Literature study assignment

Coordinators: Saba Amiri, Adam Belloum,

Note: Following are reports of the Literature study assignment part of course "Web Services and Cloud Systems"[1] given in the context of the Joint UvA-VU Computer Science program[2]. The literature assignment is worth 35% of the total course grade. Students have to read at least 17 papers and prepare a (8-10)-page report in a style of a scientific publication and give 10 mn presentation at the end of the course. The literature topics are not covered during the lectures, students use the knowledge acquired during the lectures to perform the literature study. Reports are checked for plagiarism using Trinity tool integrated in Canvas (similarity score tolerated *is max 20%).*

[1] https://studiegids.uva.nl/xmlpages/page/2022-2023/zoek-vak/vak/79525
[2] https://masters.vu.nl/en/programmes/computer-science-big-data-engineering/index.aspx

Securing the Foundation: A Comprehensive Overview of Infrastructure as Code Security, Best Practices and Novel Detection Methods

Luca Pantea
Graduate School of Informatics
University of Amsterdam
luca.pantea@student.uva.nl

Anders Nõu
Graduate School of Informatics
University of Amsterdam
anders.nou@student.uva.nl

Floris Brunet de Rochebrune
Department of Computer Science
Vrije Universiteit Amsterdam
f.d.brunetderochebrune@student.vu.nl

Abstract

Infrastructure as Code (IaC) is a vital practice in modern software development and deployment, enhancing efficiency and consistency. Despite these benefits, it introduces unique security risks that require careful management. This literature review provides a comprehensive examination of the current state of security in IaC. We explore the nature of security threats in IaC, the prevalent vulnerabilities in IaC scripts, the implications of security breaches, and the best practices for writing secure IaC scripts. Additionally, we delve into the advanced techniques for security and defect detection in IaC. This review highlights the critical importance of security considerations in IaC and proposes potential areas for future research. Our findings underscore the need for developers and organizations to adopt robust security measures and utilize advanced techniques to mitigate the potential risks associated with IaC, ensuring resilient and secure infrastructures.

1 Introduction

The growing demand for continuous delivery of code changes to meet evolving requirements has prompted numerous organizations to reconsider their development practices. One aspect that commonly halts or slows down rapid code delivery within an organisation is the tension between software development teams and operations. While the former are incentivised to deliver new releases to accommodate incoming requirements, the latter must ensure the stability of production software at all times. One particular solution the operations team use is automation logic (e.g., scripts) for the frequent deployment of middleware and applications to the production environment. However, this logic may require updates to accommodate new requirements from application releases. Moreover, these methods generally lack a design built according to software engineering practices (e.g., modularity or reusability) and often pose a risk to deployment stability [34, 25, 42, 20, 26].

One emerging framework used in DevOps to bridge the gap between engineers and operators is Infrastructure as code (IaC) [18, 31], which describes and specifies the underlying configurations behind the infrastructure of software systems, built around the fundamental principles of software engineering. By employing IaC scripts, organizations can provision and configure their development and deployment environment, set up user accounts, and manage dependencies at scale, thus allowing for a stronger collaboration between operations and software development teams, and consequently

delivering software at an increased pace. The growing interest in IaC is evident among practitioners and researchers alike [16, 37]).

Various IaC tool vendors, such as Chef[1], Puppet[2], Ansible[3] and Terraform[4] have emerged, which allow practitioners to specify configuration and dependency information as scripts, instead of manually developing and maintaining custom BASH/Perl executables across multiple cloud instances. These tools equip developers with abstractions to configure automation steps as *idempotent* [8] processes. The principle of *idempotentce* is a fundamental concept in automation and computing, ensuring consistent outcomes regardless of the number of executions.

The benefits of adopting IaC practices have been evident in information technology (IT) organizations. According to a report conducted by Enterprise Strategy Group in 2016 [27], the use of IaC scripts resulted in an average time savings of 210% for IT organizations. General Motors (GM) leveraged Chef to increase their software deployment frequency by a remarkable factor of 21 ([15]. Another notable case is the National Aeronautics and Space Administration (NASA), where the use of IaC scripts reduced their previously time-consuming patching process from multiple days to a mere 45 minutes [13]. CERN, the European Organization for Nuclear Research makes use of Puppet to effectively manage their vast infrastructure, comprising 15,000 servers and processing 2,000 TB of data every day. Puppet's deployment management capabilities have played a significant role in minimizing service disruptions and reducing deployment time at CERN [10]. These real-world examples demonstrate the tangible benefits that IaC brings to diverse IT organizations, facilitating increased efficiency, reliability, and scalability.

Despite the aforementioned advantages, IaC scripts are not immune to defects that can have severe consequences. For instance, an IaC script defect caused an outage that resulted in Amazon Web Services suffering business losses of approximately 150 million USD [17]. Similarly, other defects in IaC scripts have led to significant incidents, such as the unintentional deletion of user directories for around 270 users on cloud instances maintained by Wikimedia Commons [46]. In addition to functionality-related defects, security-related issues can also manifest in IaC scripts. These issues encompass inadvertent exposure of secrets and SSL certificates in logs, as well as hardcoding sensitive information. Notably, such security flaws are not confined to specific domains but have been observed across various projects, including blockchain [45], video game software [35], cloud management software [47], and prominent open-source projects such as Apache, Linux, and Mozilla [9, 43].

Building upon the understanding of the benefits and potential pitfalls associated with Infrastructure as Code (IaC), this study aims to delve deeper into the realm of security considerations within this domain. In light of the aforementioned defects and vulnerabilities observed in IaC scripts, it is crucial to address security concerns effectively. To accomplish this, three key research questions will guide our investigation:

- [**RQ1**]: We seek to provide a systematic overview of security considerations in Infrastructure as Code. By thoroughly examining existing literature, we aim to identify and analyze the various security aspects that must be taken into account when developing and deploying IaC scripts.
- [**RQ2**]: We propose a subset of best practices that can assist practitioners in developing secure infrastructure as code scripts. Drawing upon insights gathered from comprehensive literature studies, we will present practical guidelines and recommendations that can enhance the security posture of IaC implementations.
- [**RQ3**]: We present novel methods firmly grounded in literature for security and defect detection within the context of IaC. By leveraging advancements in security technologies and techniques, we will explore innovative approaches to identify and mitigate security vulnerabilities and defects in IaC scripts.

By addressing these research questions, this study aims to contribute to the broader understanding of security in IaC, provide actionable recommendations for practitioners, and offer perspectives on mitigating such vulnerabilities.

[1] `https://www.chef.io/`
[2] `https://www.puppet.com/`
[3] `https://www.ansible.com/`
[4] `https://www.terraform.io/`

2 Background

2.1 Definition

Infrastructure as Code (IaC) is a methodology and practice of managing and deploying infrastructure resources with code and using standard practices from software development [29]. IaC helps organizations treat the infrastructure as software. The practice involves using configuration files, scripts and automation to define and control the infrastructure configuration, deployment and management processes [28].

2.2 Benefits

Reproducibility and Consistency; Infrastructure defined as code provides the possibility of creating reproducible environments [32]. Instead of manually creating infrastructure resources, infrastructure code can be reused to replicate environments across development, testing, acceptance, and production. Therefore presenting consistency and minimizing configuration drift, which reduces the risk of deployment issues that are caused by environment discrepancies [29].

Agility and Efficiency; Leveraging infrastructure as code leads to significant benefits in terms of agility and efficiency. Defining infrastructure in code allows organizations to automate the deployment and management processes. This allows organizations to adapt quickly to business requirements and deliver fast and efficiently. IaC also greatly benefits the identification of defects and debugging due to version control [32].

Scalability and Elasticity; IaC enables organizations to rapidly upscale or downscale the infrastructure resources as required [32]. This allows teams to fully utilize the elasticity that the cloud provides. Scaling resources automatically removes manual intervention and thus reduces the risk of human errors. The dynamicity and modularity of IaC help provision infrastructure swiftly for new features and applications quickly.

Stability and Reduced Risk; Using IaC promotes stability and risk aversion [29]. Infrastructure changes can be quickly and consistently tested on development environments beforehand. This leads to fewer human errors and inconsistencies across environments. Automated infrastructure testing can also be added to the formula. Code that goes through validation and testing leads to fewer vulnerabilities and reduces the risk of downtime and disasters. In turn, the overall stability of the system improves.

Improving DevOps; Utilizing IaC helps to reduce the distance between development and operations [6]. Instead of giving the operations team full control and overview of the infrastructure, IaC shares the infrastructure responsibility with the development teams. This facilitates better communication, faster feedback loops and leads to more efficient infrastructure management.

Disaster Recovery; During incidents and catastrophic events with infrastructure, disaster recovery becomes a simple automation task [32]. The infrastructure can be rolled back, redeployed and restored due to backups and version control. Undocumented and manually configured resources and servers are no longer an issue due to IaC. This also allows organizations to do disaster recovery testing called Chaos Engineering which was pioneered in Netflix [29]. Chaos Engineering injects errors into systems to see if the systems are capable of handling the errors.

2.3 Tooling

At this point in time, there is already a multitude of tools to choose from to implement infrastructure as code. There are configuration management tools like Chef, Puppet, and Ansible that automate the provisioning, configuration, and management of the infrastructure resources [37]. In 2015, Chef and Puppet were considered the most popular infrastructure languages [19]. They provide a declarative way of defining and configuring the infrastructure as needed. Configuration management tools prove to be especially useful in managing a fleet of servers.

In addition to the configuration management tools, there are also infrastructure provisioning tools such as Terraform and AWS CloudFormation [14] [11]. Terraform provides a cloud-native way of provisioning infrastructure with declarative code. Terraform tracks the state of the infrastructure resources and compares the state to the code. During execution, Terraform updates, creates, or deletes

the resources as defined by the code. AWS CloudFormation is another provisioning tool that is dedicated to managing Amazon Web Services (AWS) cloud resources [11]. CloudFormation provides similar benefits and features to users that Terraform does with the exception that it is meant only to be used with AWS.

2.4 Examples

We show examples of four IaC languages: Chef, Puppet, Terraform and CloudFormation. Figure 1 shows Chef and Puppet snippets with the same functionality: initializing an HTTP server on two platforms [19]. Figure 2 shows an example of Terraform and CloudFormation. Both examples show how to provision a simple EC2 (Elastic Compute Cloud) instance in AWS.

```
case node[:platform]
when "ubuntu"
  package "httpd-v1" do
    version "2.4.12"
    action: install
  end
when "centOS"
  package "httpd-v2" do
    version "2.2.29"
    action: install
  end
end
```

```
case $platform{
  "ubuntu": {
    package {"httpd-v1":
      ensure => "2.4.12"
    }
}
"centOS": {
    package {"httpd-v2":
      ensure => "2.2.29"
    }
  }
}
```

Figure 1: Chef (Left) and Puppet (Right) Examples [19]

```
resource "aws_instance" "example"
{
  ami = "ami-005e54dee72cc1d00"
  instance_type = "t2.micro"
}
```

```
"ExampleEC2": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "ImageId": "ami-005e54dee72cc1d00",
    "InstanceType": "t2.micro"
  }
}
```

Figure 2: Terraform (Left) and CloudFormation (Right) Examples

3 Related Works

Our literature study showed that there is some research on the topic of infrastructure as code. Some analysed the defects in IaC, while others delved deep into specific IaC languages. There were a considerable amount of papers that gave an overview of what is IaC and explained the concept. However, on our specific topic of security in infrastructure as code, we found that there is a deficit of research. There are some papers on the topic of code smells and anti-patterns in infrastructure as code. Although, we found that a great amount of those papers were written by a select few [38], [40], [37], [36]. We found some information about the considerations, anti-patterns and methods for detecting defects in IaC. However, there was no comprehensive overview of the security that provided depth about the overall security in IaC. Overall, we discovered that papers and articles did not give an extensive summary and framework about the considerations, best practices and methods to use with infrastructure as code.

4 Methodology

This study employs a two-pronged approach, combining a systematic literature review and an empirical analysis of internet artifacts, to explore the security considerations pertinent to Infrastructure as Code (IaC). The process can be divided into four main steps.

1. Exploratory search for IaC security literature and Internet artifacts; We perform a systematic search for pertinent peer-reviewed articles within databases such as IEEE Xplore, ACM Digital Library, SpringerLink, and also employ Google Scholar and Google Search for a broader reach. Our

search string includes keywords such as "infrastructure as code security", "infrastructure as code security practices", "puppet scripts security", "chef cookbook security", and "terraform code security". We collect the top 25 results from each search engine for every query string, creating an initial pool of potentially relevant sources.

2. Filtering of literature and artifacts; To establish a corpus of high-quality, relevant literature, we then apply a series of rigorous filtering criteria. This includes excluding non-English results, eliminating duplicate entries, and retaining only those sources directly discussing IaC security, security management techniques, or secret management best practices. The final corpus of 41 works comprises academic papers and Internet artifacts like blog posts and articles (showcased partly in the introduction, and analysed in the Results section).

3. Synthesis of the literature; Following the collection and filtration process, the remaining artifacts are scrutinized to identify common themes, best practices, and recurring patterns. Through a thematic analysis approach, we categorize and summarize findings to distil a set of effective practices that have been highlighted in the literature. This synthesis informs our understanding of prevailing security considerations in IaC and contributes to the formulation of recommendations for secure IaC implementations.

4. Enumeration of methods aligning with best practices; Lastly, we catalogue tools and methods that are aligned with the synthesized best practices. This includes solutions for identifying and mitigating security vulnerabilities in IaC scripts. This step aids in translating the synthesized best practices into actionable strategies for practitioners aiming to enhance the security of their IaC deployments.

This dual-pronged approach, combining rigorous academic research with insights from real-world applications, offers a comprehensive view of IaC security, allowing for robust conclusions and actionable recommendations.

5 Results

5.1 [RQ1] Security Implications in IaC Development and Deployment

Nature of Security Threats in IaC; Infrastructure as Code (IaC) enhances agile software development but also ushers in unique security threats. TrendMicro highlights the risks of misconfiguration, where improper settings lead to security breakdowns, the complexity of managing drift in immutable infrastructure, issues in handling confidential data like passwords and API keys, the necessity of stringent access management policies, and the requirement for diligent logging and monitoring [44]. Simultaneously, OWASP proposes several countermeasures: using standard security plug-ins in IDEs for early threat detection, initiating threat modelling early in the development cycle for enhanced visibility and flexibility, securely managing secrets, utilizing version control for tracking IaC changes, and performing static analysis to identify potential risks and misconfigurations.

Prevalent Vulnerabilities in IaC Scripts; Several academic articles have reported on the prevalent vulnerabilities in IaC scripts. One such article is “Security Controls in Infrastructure as Code” by Almuairfi and Alenezi [5]. The authors conducted a study of security smells in IaC scripts and identified seven security smells through qualitative analysis of 1,726 IaC scripts. The security smells include hard-coded secrets, weak permissions, and insecure network configurations.

Another article is “The ‘as code’ activities: development anti-patterns for infrastructure as code” by Knauss et al. [22]. The authors identified development anti-patterns that relate to defective IaC scripts. These anti-patterns include poor documentation, lack of testing, and poor code quality.

Finally, “Source code properties of defective infrastructure as code scripts” by Knauss et al. [21] identified 12 source code properties that correlate with defective IaC scripts. These properties include long files, high complexity, and low cohesion.

Impact and Implications of Security Breaches; Security breaches in IaC can have serious consequences for organizations. According to Neharika and Lennon [30], security attacks are rapidly increasing and can result in heavy fines when appropriate measures for security are not taken. Provisioning infrastructure using manual configuration can also be a difficult task as it involves multiple steps. The authors investigated securely provisioning infrastructure automatically using source code analysis tool, container security tool, and IaC tools. They showed that source code and containers

can be scanned for vulnerabilities, and when critical vulnerabilities are not found, the infrastructure can be automatically provisioned using Terraform script. The authors observed that implemented systems can be scanned for vulnerabilities in source code and containers provisioned automatically using secure IaC script.

In addition, cyber risk management is an important aspect of cybersecurity. According to a systematic review of data by Kshetri et al. [23], cyber risk management is essential for organizations to protect their data from cyber attacks. The authors analyzed the extant academic and industry literature on cybersecurity and cyber risk management with a particular focus on data availability. They found that cyber risk management is an important aspect of cybersecurity and that organizations should take proactive measures to protect their data from cyber attacks.

Security Management in IaC; IaC security is the practice of securing infrastructure that is managed using IaC. This can include measures to secure the IaC codebase itself, as well as the infrastructure resources that are managed using IaC [1]. According to OWASP, security in IaC contexts is typically managed by developers who are responsible for writing the code that defines the infrastructure[2]. However, security teams may also be involved in reviewing and approving the code before it is deployed. Automation plays a critical role in IaC security by enabling security checks to be performed automatically during the development process. This helps to identify potential security issues early on and ensures that security is integrated into the development process from the beginning.

Security should be integrated throughout the entire Infrastructure as Code life cycle, from development to deployment and beyond. According to TechTarget, IaC can help organizations achieve better security by providing greater visibility into their infrastructure and enabling them to more easily identify and remediate vulnerabilities[3]. By treating infrastructure as code, organizations can apply the same versioning techniques used for software development teams, which can help ensure that changes are tracked and auditable.

5.2 [RQ2] Best Practices for Secure IaC Scripts

As the popularity of using Infrastructure as Code rises, it is vital that the security concerns of the IaC are taken into attention. If the security of the scripts is not considered, organizations could introduce vulnerabilities in their infrastructure. This section highlights the best practices for writing secure IaC scripts to maintain secure, durable and resilient infrastructure. It is also important to note that we cover the best practices for both the code itself and also some security considerations for what is deployed and provisioned by the IaC.

Principle of least privilege; Analysis showed that the principle of least privilege is sometimes violated [38]. This means that the IaC is provisioning infrastructure that gives the resources or users too much access, often administration access. Instead, applying the zero-trust model for the services is more secure because the application only has as few permissions as needed and only to the resources it needs to access [29].

Use a secret management tool; Researchers found that more often than not, secrets were hard-coded and exposed in the Git history [4], [38]. Usernames, passwords, API keys etc are often exposed which causes vulnerabilities and might lead to security breaches. Instead of using hard-coded secrets, it is recommended to use secret management tools where the secrets could be fetched from and injected into the code [36]. Some examples of secret management tools are HashiCorp's Vault [5], AWS Secrets Manager [6] and GCP Secret manager [7]

Fine-grained access control; Developers might often take shortcuts when implementing access controls for servers, databases and other resources [38]. For the initial testing and validation process it is easy to take the path of allowing access from everywhere to not deal with connectivity issues. However, when this security anti-pattern stays in the system permanently, security concerns rise up. Instead of defining access to be allowed from `0.0.0.0/0` (everywhere) for servers and databases, specific IP ranges of private networks should be utilized instead.

Testing; Performing continuous testing and delivery is one of the core practices of Infrastructure as Code. For secure, resilient and durable infrastructure, testing is a vital cornerstone for achieving that.

[5] `https://www.vaultproject.io/`

[6] `https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html`

[7] `https://cloud.google.com/secret-manager`

It is reasonable to test in addition to many other aspects of the infrastructure, the security of the code. Such as tests for asserting the status of ports, user access and overall resource access [29].

Other considerations; In addition to the aforementioned practices, it could also be important to keep in mind other security concerns that could lead to serious threats. Such concerns include the employment of TLS to prevent main-in-the-middle attacks [38]. In addition to using a secret management tool to avoid hard-coding secrets, it is important to obfuscate secrets in the logs as well [4]. The final concern is the use of weak cryptography algorithms when for example setting up passwords or SSH connections. Using weak algorithms could lead to collision attacks and might open up your system to intruders [38].

5.3 [RQ3] Advanced techniques for identifying and mitigating security vulnerabilities in IaC scripts

This section aims to give an overview of the recent research efforts focused on the development of techniques for the analysis and detection of security anomalies in IaC scripts.

Model-Based Approach for Assessing Security-Related Practices; The widespread adoption of IaC has revolutionized software deployments, but verifying adherence to architectural requirements remains a challenge. In the paper titled "Assessing Architecture Conformance to Security-Related Practices in Infrastructure-as-Code-Based Deployments" [33], the authors investigate security-related practices in deployment architectures managed using IaC scripts. Their proposed model-based approach utilizes technology-independent metrics tied to architectural design decisions in IaC-based deployments, allowing developers to measure conformance to security practices such as observability, access control, and traffic control.

To evaluate the effectiveness of their approach, the authors employ 21 IaC deployment models. Initially, they model the essential aspects of architectural decision options using a minimal set of deployment model elements, automatically extracting them from the IaC scripts. Subsequently, a range of metrics is defined to cover various decision options, with the generated models serving as a benchmark. Ordinal regression analysis is then employed to derive a prediction model. The results demonstrate the efficacy of the proposed metrics, accurately predicting the ground truth assessment with a high degree of accuracy. The significance of this approach lies in its ability to facilitate the continuous assessment of deployment models, considering the influence of continuous delivery practices.

Detection of Security Smells with SLIC; Security smells encompass patterns that serve as indicators of vulnerabilities within a system, making them potentially exploitable. Just like any other software system, Infrastructure as Code (IaC) is also susceptible to security problems. Rahman et al. [38] carried out a study into coding practices that could potentially compromise security within Puppet scripts. The study involved scrutinizing 1,726 Puppet scripts sourced from Mozilla, OpenStack, and Wikimedia projects. Through descriptive coding, the authors identified 7 specific "security smells", which are common indicators of possible security issues, using, which often occur beyond the context of Infrastructure as Code (e.g. "hard-coded secret", "admin by default"). To detect these security smells in Puppet scripts, the authors develop and evaluate SLIC, a rule-based method of detecting security smells based on pre-defined string patterns.

The authors tested SLIC on a sample of 140 Puppet scripts containing manually labelled as containing code smells, where it showed increased precision and recall rates. Further examination was carried out on a larger scale, running SLIC on 15,232 Puppet scripts collected from GitHub, Mozilla, OpenStack, and Wikimedia repositories. The 'hard-coded secret' smell was found to be most prevalent, appearing in 21.9%, 9.9%, 24.8%, and 17.0% of scripts from the respective sources. The authors engaged industry professionals by submitting 1,000 bug reports related to identified instances of security smells, resulting in a 21.2% response rate. Among the responded reports, there was a consensus of 69.8% on the identified bugs, with the highest agreement observed for the "use of weak cryptography algorithms" (84.6%) and "use of HTTP without TLS" (over 75%).

Expansion to Ansible and Chef Scripts with SLAC; A follow-up study by Rahman et al. [39] expands the investigation of security smells to Ansible and Chef scripts, building upon the previous research. They identify two additional smells and develop SLAC, a tool for detecting these smells along with the ones identified in the earlier study. Evaluations demonstrate SLAC's high precision and recall in Ansible and Chef scripts (0.9 and 1.0, respectively). Empirical studies on thousands of

scripts reaffirm the prevalence of "hard-coded secret" (22.4% Ansible scripts from OpenStack and 6.8% Chef scripts on GitHub) in [38] and other common smells, such as "suspicious comment" and "use of HTTP without TLS". A practitioner survey reveals agreement on security smells, although the response rate was relatively low (9.4%), which should be considered when interpreting the findings.

GLITCH for Polyglot Security Smell Detection; The tools mentioned above are of great value for practitioners, as they span a wide range of security smells and are built for the prevalent IaC technologies (Puppet, Ansible and Chef). However, they are implemented separately (prone to duplication), language-dependent and lack flexibility in introducing a new smell (one would have to implement the logic for both technologies). To address this, the authors of "GLITCH: Automated Polyglot Security Smell Detection in Infrastructure-as-Code" [41] propose a language-agnostic tool that provides consistent security checks and smell detection for IaC scripts.

The proposed method achieve state-of-the-art (SOTA) performance by transforming IaC scripts (Ansible, Chef or Puppet) into a new intermediate representation that supports the detection of nine security smells. The framework was tested on a dataset of nearly 200,000 scripts and over 12 million lines of code. It operates based on security rules like "admin by default" and "hard-coded secret". When evaluated on this large corpus of IaC scripts, GLITCH outperformed existing tools (like SLAC) in terms of precision and recall and demonstrated flawless results for rules such as "admin by default". The lowest precision was found to be 42% for the "hard-coded secret" rule. The method displayed significant improvements in both precision and recall for Puppet, Ansible, and Chef scripts. For Puppet, precision and recall increased by 8% and 13% respectively, while Ansible saw improvements of 10 and 8 points. The most noticeable enhancement was for Chef, with precision and recall increasing by 28 and 26 percentage points respectively. Additionally, GLITCH also provided speedups over SLIC and SLAC, ranging from 9x to 32x.

Defect Prediction using RADON; Defective Infrastructure as Code (IaC) scripts can lead to severe outcomes for software development organizations, such as monetary loss, security breaches and from service disruptions. Predicting such failure-prone scripts is critical as it aids developers in prioritizing their examination efforts and allocating resources effectively. In their article "Within-Project Defect Prediction of Infrastructure-as-Code Using Product and Process Metrics," Dalla Palma et al. [12] introduce RADON, a Machine Learning framework designed to predict IaC defects. RADON utilizes a variety of code, IaC-specific, and process metrics to perform tasks such as repository crawling, metrics gathering, model creation, and assessment.

The authors conducted a thorough empirical study on Ansible code to evaluate RADON's efficacy in IaC defect prediction. They compared five ML methods for defect prediction in IaC scripts: decision tree, logistic regression, naive Bayes, random forest, and support vector machine. Using a publicly available dataset of Ansible-based IaC scripts from GitHub, they trained these models and assessed their predictive abilities. The study found that the random forest model was the most effective, and models using IaC-specific metrics (such as TEXTENTROPY, NUMKEYS) performed better than those relying on other metric sets. Moreover, the study identified certain IaC-oriented metrics that optimize prediction performance. These insights provide a guideline for choosing suitable prediction models based on project-specific features and inform the development of prediction models for IaC scripts. Despite the focus on Ansible, the authors believe the predictive models can be easily transferred to other languages like Chef and Puppet, as several of the most influential features are general-purpose metrics.

Detection of Linguistic Anti-Patterns with DeepIaC; Finally, linguistic anti-patterns, which are recurrent poor practices causing inconsistencies in the naming, documentation, and implementation of an element, not only obstruct the readability, understandability, and maintainability of source code but also pose security challenges. In response to this security-sensitive problem, N. Borovits and his team [7] devised an innovative solution, DeepIaC, which is an automated tool that employs word embeddings and deep learning techniques, specifically Convolutional Neural Networks [24], to classify scripts based on the presence of these linguistic anti-patterns. The tool is trained on a dataset of scripts, which have been purposefully modified with bugs.

DeepIaC's effectiveness was empirically trained and evaluated using a set of 18,286 scripts sourced from 38 GitHub repositories, with artificially-inserted bugs. The tool exhibited strong performance with an accuracy range of 0.785 to 0.915. Its standout performance was in detecting inconsistency in the file module, where accuracy peaked at 0.915 and F1 scores for both inconsistent and consistent classes exceeded 0.9.

6 Discussion

The literature review presented in the previous sections has identified key security issues and best practices related to Infrastructure as Code (IaC) development and deployment. This section seeks to discuss these findings, particularly focusing on their implications for future research and practical applications in the field.

6.1 Security Implications in IaC

The study of IaC has revealed its potential for bolstering agile software development but also the associated unique security threats. Misconfiguration, drift in immutable infrastructure, insecure handling of confidential data, lax access management policies, and weak logging and monitoring policies pose serious challenges to security [44]. Several studies have identified prevalent vulnerabilities in IaC scripts, which include hard-coded secrets, weak permissions, insecure network configurations, poor documentation, lack of testing, and poor code quality [5, 22, 21].

These security vulnerabilities can lead to severe implications, including substantial financial losses and reputational damage [30]. It's therefore critical to manage IaC security effectively, involving both developers and security teams, to perform regular checks and apply preventive measures [1, 2, 3].

6.2 Best Practices for Secure IaC Scripts

Despite the potential risks, IaC has immense benefits, providing developers with more control and visibility into their infrastructure. To leverage these benefits while mitigating the associated security risks, certain best practices should be followed. These practices involve applying the principle of least privilege [38], using secret management tools [36], implementing fine-grained access control [38], continuous testing [29], and other security considerations like the use of TLS and avoiding weak cryptography algorithms [38].

These best practices present a roadmap for developers and organizations to write and maintain secure IaC scripts, hence promoting a secure, durable, and resilient infrastructure.

6.3 Advanced Techniques for Security and Defect Detection in IaC

Apart from following the best practices, researchers have also proposed advanced techniques for detecting defects and enhancing security in IaC. The reviewed papers take on different angles in addressing security within the context of Infrastructure as Code (IaC), with unique results and varying levels of effectiveness. The model-based approach, presented in the first discussed paper, capitalizes on architectural design decisions in IaC deployments. The paper successfully argues for a model-based method that leverages technology-independent metrics tied to architectural design decisions. It further consolidates this argument with empirical evidence, employing 21 IaC deployment models to demonstrate the effectiveness of their method. Notably, the metrics predicted the ground truth assessment with a high degree of accuracy. The method's strength resides in its potential to continuously evaluate IaC-based deployments, thus aligning with the core principles of DevOps culture – continuous integration and continuous deployment. However, one potential drawback is that it assumes the correct and complete extraction of model elements from IaC scripts, which may not always be feasible, particularly when dealing with large, complex, or poorly documented scripts.

The work of Rahman et al. encapsulated in two sequential papers, explores the identification and detection of 'security smells' in IaC scripts. Their first paper, "*The Seven Sins*: Security Smells in Infrastructure as Code Scripts" takes a focused view, investigating the coding practices that could potentially compromise security within Puppet scripts. By defining and detecting seven distinct security smells, the paper introduces a conceptually new and practical means of revealing potential security issues. The second paper expands this approach to Ansible and Chef scripts with SLAC. The follow-up research not only validates the previous work but also broadens its scope, thereby enhancing practical relevance. A potential limitation of their approach is its reliance on predefined smell patterns, potentially limiting its effectiveness against novel or less typical security threats.

The fourth study, "GLITCH: Automated Polyglot Security Smell Detection in Infrastructure as Code", takes a significant step forward by proposing a language-agnostic tool to detect security smells. The strength of this approach lies in its versatility – GLITCH's intermediate representation allows for more consistent security checks across multiple IaC technologies (Puppet, Ansible, and Chef). Moreover,

the tool's performance surpasses previous solutions, showing improved precision and recall for all three languages. However, a key consideration is that it still relies on detecting predefined security smells, much like SLIC and SLAC, which could limit its effectiveness against non-standard or newly emerging threats.

In the fifth analysed paper, the authors take a different approach to IaC security by employing machine learning to predict defective IaC scripts. The strength of RADON lies in its ability to effectively prioritize and manage the allocation of resources by providing predictive insights into potential defects. It also gives developers actionable intelligence on which areas of their scripts need attention. However, the predictive performance of the model could be compromised by the quality and representativeness of the training data used or the inherent complexity of the defect prediction problem.

The final paper, "Detection of Linguistic Anti-Patterns with DeepIaC," adds a new dimension to the security analysis of IaC scripts. The proposed tool, DeepIaC, harnesses the power of word embeddings and deep learning to identify and classify scripts based on the presence of linguistic anti-patterns. This approach is quite potent in that it can detect less obvious, yet consequential vulnerabilities that stem from linguistic inconsistencies in IaC scripts. However, the tool's effectiveness in real-world scenarios may be limited by the nature of its training set, which was artificially modified with bugs.

In essence, each study offers valuable insights into mitigating security concerns in IaC contexts, each with unique strengths and potential limitations. However, the varied methodologies highlight the multifaceted nature of IaC security, suggesting that a combined, comprehensive approach may be more effective in achieving robust and enduring security practices.

6.4 Future Directions

The ever-increasing reliance on IaC to manage modern cloud-based infrastructures necessitates further research in this domain. Future research should delve deeper into understanding and modelling the security risks associated with IaC, developing novel techniques for automatic vulnerability detection, and creating tools that can assist developers in writing secure IaC scripts.

Moreover, future work should also aim to bridge the gap between research and practice by proposing industry-relevant methodologies that ensure security in IaC without hampering the agility and productivity it offers. Industry-academia collaborations can play a significant role in this regard by combining practical industry insights with rigorous academic research methods to create more robust, scalable, and secure IaC solutions.

7 Conclusion

In this paper, we have examined the state of the art in Infrastructure as Code (IaC) security, identifying the most prevalent security threats and the implications of these risks. We have also highlighted several best practices and advanced techniques that are essential for enhancing security in IaC scripts. Our literature review emphasizes the importance of maintaining a balance between leveraging the benefits of IaC and mitigating the associated security risks.

IaC has emerged as a powerful tool in managing modern cloud-based infrastructures. However, the potential security vulnerabilities in IaC scripts can have severe implications if not properly managed. It's imperative for developers and organizations to adopt robust security measures, such as the principle of least privilege, fine-grained access control, continuous testing, and use of secret management tools.

The use of advanced techniques like static code analysis and machine learning for anomaly detection has shown promise in enhancing security in IaC. Future research should focus on building upon these techniques and developing tools that can assist in writing secure IaC scripts. Moreover, collaborations between academia and industry can greatly contribute to the development of practical, robust, and secure IaC solutions.

In conclusion, as the reliance on IaC continues to grow, it is crucial for all stakeholders to prioritize security in IaC development and deployment, thereby ensuring the resilience and integrity of the infrastructures they manage.

Bibliography

[1] Infrastructure as code (iac) security: Securing top 5 iac platforms - aqua. https://www.aquasec.com/cloud-native-academy/devsecops/infrastructure-as-code-iac/. Accessed: 2023-06-011.

[2] Infrastructure as code security - owasp cheat sheet series. https://cheatsheetseries.owasp.org/cheatsheets/Infrastructure_as_Code_Security_Cheat_Sheet.html. Accessed: 2023-06-01.

[3] The security benefits of using infrastructure as code. https://www.techtarget.com/searchsecurity/tip/The-security-benefits-of-using-infrastructure-as-code. Accessed: 2023-06-01.

[4] Effat Farhana Akond Rahman, Chris Parnin, and Laurie Williams. Gang of eight: A defect taxonomy for infrastructure as code scripts. in 2020 ieee/acm 42nd international conference on software engineering (icse). acm, seoul south korea, 752–764, 2020.

[5] Sadiq Almuairfi and Mamdouh Alenezi. Security controls in infrastructure as code. *Computer Fraud & Security*, 2020(10):13–19, 2020.

[6] Matej Artac, Tadej Borovssak, Elisabetta Di Nitto, Michele Guerriero, and Damian Andrew Tamburri. Devops: Introducing infrastructure-as-code. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 497–498, 2017.

[7] Nemania Borovits, Indika Kumara, Parvathy Krishnan, Stefano Dalla Palma, Dario Di Nucci, Fabio Palomba, Damian A. Tamburri, and Willem-Jan van den Heuvel. Deepiac: Deep learning-based linguistic anti-pattern detection in iac, 2020.

[8] Mark Burgess. Testable system administration. *Commun. ACM*, 54(3):44–49, mar 2011.

[9] Gemma Catolino, Fabio Palomba, Andy Zaidman, and Filomena Ferrucci. Not all bugs are the same: Understanding, characterizing, and classifying the root cause of bugs, 2019.

[10] CERN. Key facts and figures – cern data centre. https://information-technology.web.cern.ch/sites/default/files/CERNDataCentre_KeyInformation_01June2018V1.pdf, 2018. [Online; accessed 2-June-2023].

[11] Peter Dalbhanjan. Overview of deployment options on aws. *Amazon Whitepapers*, 2015.

[12] Stefano Dalla Palma, Dario Di Nucci, Fabio Palomba, and Damian A. Tamburri. Within-project defect prediction of infrastructure-as-code using product and process metrics. *IEEE Transactions on Software Engineering*, 48(6):2086–2104, 2022.

[13] Jonathan Davila. Nasa case study. https://fiercesw.com/wp-content/uploads/2016/01/NASA-Case-Study-Ansible.pdf, 2016. [Online; accessed 2-June-2023].

[14] Leonardo Rebouças De Carvalho and Aleteia Patricia Favacho de Araujo. Performance comparison of terraform and cloudify as multicloud orchestrators. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 380–389. IEEE, 2020.

[15] Jeanne Gu. Gm - extract, transform, and load platform as a service. https://www.chef.io/docs/cheflibraries/stories/chef_gm_etl_platform_as_a_service.pdf, 2019. [Online; accessed 2-June-2023].

[16] MIchele Guerriero, Martin Garriga, Damian A. Tamburri, and Fabio Palomba. Adoption, support, and challenges of infrastructure-as-code: Insights from industry. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 580–589, 2019.

[17] Rebecca Hersher. Amazon and the $150 million typo. https://www.npr.org/sections/thetwo-way/2017/03/03/518322734/amazon-and-the-150-million-typo, 2017. [Online; accessed 2-June-2023].

[18] Michael Httermann. *DevOps for Developers*. Apress, USA, 1st edition, 2012.

[19] Yujuan Jiang and Bram Adams. Co-evolution of infrastructure and source code - an empirical study. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 45–55, 2015.

[20] Muhammad Shoaib Khan, Abudul Wahid Khan, Faheem Khan, Muhammad Adnan Khan, and Taeg Keun Whangbo. Critical challenges to adopt devops culture in software organizations: A systematic review. *IEEE Access*, 10:14339–14349, 2022.

[21] Eric Knauss, Thomas Zimmermann, and Kurt Schneider. Source code properties of defective infrastructure as code scripts. *Journal of Systems and Software*, 157:1–14, 2019.

[22] Eric Knauss, Thomas Zimmermann, and Kurt Schneider. The 'as code' activities: development anti-patterns for infrastructure as code. *Empirical Software Engineering*, 25(6):4085–4118, 2020.

[23] Nir Kshetri. Cyber risk and cybersecurity: a systematic review of data availability. *Journal of Information Security*, 13(1):1–20, 2022.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. A survey of devops concepts and challenges. *ACM Comput. Surv.*, 52(6), nov 2019.

[26] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. A survey of DevOps concepts and challenges. *ACM Computing Surveys*, 52(6):1–35, nov 2019.

[27] Mike Leone. The economic benefits of puppet enterprise. cost-effectively automating the delivery, operation, and security of an it infrastructure. `https://www.esg-global.com/hubfs/pdf/EVV-Report-Ex2.pdf`, 2016. [Online; accessed 2-June-2023].

[28] Kief Morris. *Infrastructure as code: managing servers in the cloud*. " O'Reilly Media, Inc.", 2016.

[29] Kief Morris. *Infrastructure as code*. O'Reilly Media, 2020.

[30] Keerthi Neharika and Ruth G Lennon. Investigations into secure iac practices. In *Lecture Notes in Networks and Systems*, volume 448, pages 301–310. Springer, 2022.

[31] Stephen Nelson-Smith. *Test-Driven Infrastructure with Chef: Bring Behavior-Driven Development to Infrastructure as Code*. O'Reilly Media, Inc., 2013.

[32] Stephen Nelson-Smith. *Test-Driven Infrastructure with Chef: Bring Behavior-Driven Development to Infrastructure as Code*. " O'Reilly Media, Inc.", 2013.

[33] Evangelos Ntentos, Uwe Zdun, Ghareeb Falazi, Uwe Breitenbücher, and Frank Leymann. Assessing architecture conformance to security-related practices in infrastructure as code based deployments. In *2022 IEEE International Conference on Services Computing (SCC)*, pages 123–133, 2022.

[34] Chris Parnin, Eric Helms, Chris Atlee, Harley Boughton, Mark Ghattas, Andy Glover, James Holman, John Micco, Brendan Murphy, Tony Savor, Michael Stumm, Shari Whitaker, and Laurie Williams. The top 10 adages in continuous deployment. *IEEE Software*, 34(3):86–95, 2017.

[35] Luca Pascarella, Fabio Palomba, Massimiliano Di Penta, and Alberto Bacchelli. How is video game development different from software development in open source? In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 392–402, 2018.

[36] Akond Rahman, Farhat Lamia Barsha, and Patrick Morrison. Shhh!: 12 practices for secret management in infrastructure as code. In *2021 IEEE Secure Development Conference (SecDev)*, pages 56–62. IEEE, 2021.

[37] Akond Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams. A systematic mapping study of infrastructure as code research. *Information and Software Technology*, 108:65–77, 2019.

[38] Akond Rahman, Chris Parnin, and Laurie Williams. The seven sins: Security smells in infrastructure as code scripts. in 2019 ieee/acm 41st international conference on software engineering (icse). *IEEE, 164ś175*, 2019.

[39] Akond Rahman, Md. Rayhanur Rahman, Chris Parnin, and Laurie Williams. Security smells in ansible and chef scripts: A replication study, 2020.

[40] Akond Rahman and Laurie Williams. Source code properties of defective infrastructure as code scripts. *Information and Software Technology*, 112:148–163, 2019.

[41] Nuno Saavedra and João F. Ferreira. Glitch: Automated polyglot security smell detection in infrastructure as code, 2022.

[42] Mali Senapathi, Jim Buchan, and Hady Osman. Devops capabilities, practices, and challenges: Insights from a case study. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, EASE '18, page 57–67, New York, NY, USA, 2018. Association for Computing Machinery.

[43] Lin Tan, Chen Liu, Zhenmin Li, Xuanhui Wang, Yuanyuan Zhou, and Chengxiang Zhai. Bug characteristics in open source software. *Empirical Softw. Engg.*, 19(6):1665–1705, dec 2014.

[44] TrendMicro. Infrastructure as code security risks and how to avoid them. *TrendMicro*, 2020.

[45] Zhiyuan Wan, David Lo, Xin Xia, and Liang Cai. Bug characteristics in blockchain systems: A large-scale empirical study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 413–424, 2017.

[46] Wikitech. Incidents/2017-01-18 labs — wikitech,. `https://wikitech.wikimedia.org/w/index.php?title=Incidents/2017-01-18_Labs&oldid=1965859`, 2022. [Online; accessed 2-June-2023].

[47] Wei Zheng, Chen Feng, Tingting Yu, Xibing Yang, and Xiaoxue Wu. Towards understanding bugs in an open source cloud management stack: An empirical study of openstack software bugs. *Journal of Systems and Software*, 151:210–223, 2019.

Serverless Computing and Function as a Service: A Literature Study

Adam Swift
Department of Computer Science
University of Amsterdam
Science Park 904, 1012WX Amsterdam
a.k.swift@student.vu.nl

Sebastian Skalski
Department of Computer Science
University of Amsterdam
Science Park 904, 1012WX Amsterdam
sebastian.skalski@student.uva.nl

Maciej Kozub
Department of Computer Science
University of Amsterdam
Science Park 904, 1012WX Amsterdam
maciej.kozub@student.uva.nl

Abstract

Serverless Computing and Function as a Service (FaaS) have risen to prominence in the last decade due to the wealth of FaaS offerings from public cloud companies including Amazon, Google and Microsoft. This Literature Study explores how Serverless architectures impact the scalability and resiliency of applications, as well as techniques that can be leveraged to design scalable and resilient Serverless applications. We investigate studies such as the Survey on Serverless Computing[12] and A Fault-Tolerance Shim for Serverless Computing[29], and discuss the utility of FaaS in different contexts. We conclude that although serverless architectures provide many benefits to scalability and resiliency, there are also drawbacks that decrease their utility in certain use cases, such as in stateful applications.

1 Introduction

Serverless Computing and Function as a Service (FaaS) are technologies that have been rising in popularity in recent years, as seen by the market size and Google Trends. Serverless architectures allow developers to focus on writing code, as the management of infrastructure is abstracted and operational costs depend on actual use. Thus, using a Severless approach often results in application development that is faster, more cost-effective, and more flexible than a traditional server-based approach. Consequently, Serverless Computing and FaaS has become an important topic in the current technology landscape.

In this literature study we aim to answer two connected research questions: 1. How do serverless architectures impact application scalability and resiliency? and 2. How do we design serverless applications that are scalable and resilient?

Scalability and Resiliency are important factors in the development of the majority of applications. Developers aspire to create scalable applications such that they can handle increasing amounts of users and data when demand increases. If applications scale badly, higher amounts of users could cause performance to decrease (or halt), negatively affecting the users' experience. Similarly, application resiliency is crucial to minimise the effect of failures and to ensure the application can continue to function smoothly. Scalability and Resiliency in Serverless computing has been investigated in the past but often with limited detail, especially in the case of Resiliency, as most reviews have preferred to analyse other factors of Serverless architectures such as their Performance and Latency, Cost

optimisation, or Security. Therefore, we have decided to focus on these areas and we assert that our research questions are of high significance.

We have structured the remainder of our study as follows: In Section 2 we give some background on the origins of Serverless Computing and FaaS, and detail some of the key concepts of Serverless architectures. Additionally, we discuss some of the main benefits and challenges of these architectures. We explore studies and literature regarding scalability in Section 3, including a case study of the Toyota Connected Serverless Architecture. Then, we present the current literature regarding serverless computing resiliency in Section 4. In Section 5, we give more detail on specific design techniques and actionable advice that can be employed to develop resilient and scalable serverless applications. In Section 6, We give our opinion on the research and discuss how applicable serverless computing is to stateful applications. Lastly, We conclude our findings in Section 7.

2 Background & Concepts

Cloud computing has significantly evolved over the years, expanding its capabilities and transforming how businesses operate and how applications are developed. Initially operating as an Infrastructure as an Service (IaaS) [22] model, which provides basic compute resources, such as virtual machines, storage, and networks, as services over the Internet. IaaS provides users with the highest level of flexibility and control over their infrastructure, but it also requires users to manage and maintain their own operating systems, middleware, and applications. Examples of IaaS are Amazon EC2, Azure VM and Google Compute Engine. Next cloud services providers started introducing Platform as a Service (PaaS) [11], to further abstract the underlying infrastructure. PaaS provides a platform that includes both hardware and software tools over the Internet. This model eliminates the need for users to manage the underlying infrastructure and allows them to focus on the deployment and management of their applications. PaaS examples include AWS Elastic Beanstalk, Google App Engine and Microsoft Azure Web Services. Currently cloud computing is embracing Serverless Computing and Function as a Service (FaaS) [10]. Each phase of this progression has sought to enhance usability, decrease costs, and augment the degree of abstraction presented to the developer.

2.1 Serverless Computing and Function as a Service (FaaS)

FaaS is the most recent step in cloud computing's evolutionary journey. In a serverless model, the cloud provider takes over all the operational responsibilities, allowing developers to focus solely on writing and deploying code. The term "serverless" does not mean that there are no servers, but rather that the management of these servers is entirely handled by the cloud provider. The most notable examples of FaaS include AWS Lambda, Google Cloud Functions, Apache OpenWhisk and Azure Functions [27]. *The Rise of Serverless Computing* [7] formally defines serverless computing as a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running.

The FaaS service model started gaining a lot of attention after the release of AWS Lambda in 2014, which started the overall serverless trend [31]. Functions deployed to FaaS platforms are typically event-driven, stateless, and often have short execution time. Developers outsource the maintenance efforts to the corresponding platform provider. As a consequence, functions are automatically scaled without any imposed limits on the amount of new instances [31]. According to [7], due to its simplicity and economical advantages, serverless computing is gaining popularity as reported by the increasing rate of the "serverless" search term by Google Trends.

FaaS isn't the sole component of serverless computing. There's also a set of specialized services provided by cloud vendors, encompassing areas such as storage (for example, AWS S3, AWS Elasticache, DynamoDB, Cloud Firestore, Cloud Pub/Sub), machine learning (for instance AWS SageMaker), and data analytics (such as AWS EMR, Google BigQuery, and Google Cloud Dataflow). These services fall under the Backend as a Service (BaaS) category [23]. Although many of these services were already in existence when FaaS was just beginning to be accessible to the public, they can be viewed as serverless services with a specific focus, given that they similarly free developers from managing the underlying hardware resources. The entirety of serverless computing is a blend of the generic FaaS platform and the supplementary services provided by BaaS. For example, lambda functions frequently employ BaaS storage services such as AWS S3 for storing their input and output data. Consequently, serverless computing is considered as the fusion of FaaS and BaaS [23].

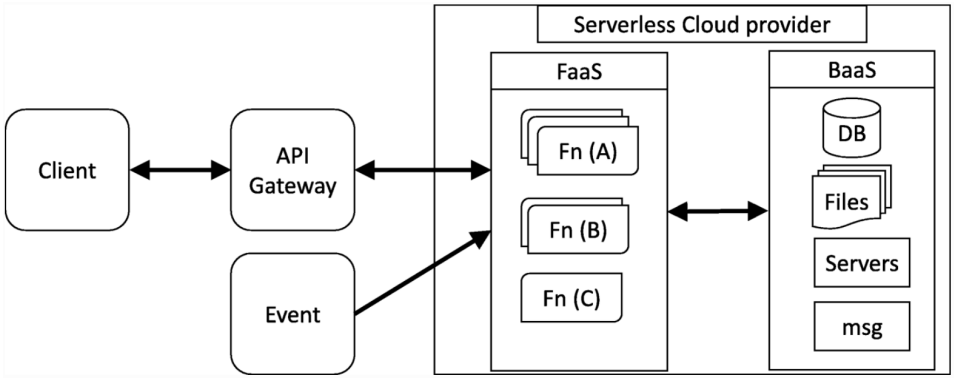


Figure 1: Example of serverless back-end of CSP [12]

2.2 Benefits

Serverless architectures offer several key benefits.

1. They provide a high degree of scalability, as the cloud provider can instantly provision more resources to handle increased demand. Thus, they benefit from cloud providers' economy of scale on resource utilization.
2. They allow for precise billing, as users only pay for the compute time they consume which is typically less than the cost of running dedicated servers. The pay as you use concept is one of the key aspects of serverless computing. Depending on the exact scenario, building in serverless often leads to a reduction in cost, particularly on variable work loads. However, in some cases, such as on some stable workloads, the costs can be greater [12].
3. One of the unique and key benefits of serverless computing is the capability to scale to zero instances when there are no requests hitting the service. This means that there is no cost incurred until the application is invoked by an event.
4. Serverless architectures also have the potential to improve developer productivity, as developers can focus on writing code without worrying about infrastructure management.

2.3 Challenges

There a several drawback to serverless computing and FaaS.

1. They introduce new complexities in areas such as testing, monitoring, and debugging. Additionally [12] observed a lack of solutions for testing, debugging and versioning FaaS-based projects, especially for separately testing, debugging and versioning of FaaS-based applications from the functions used to implement them.
2. Serverless platforms have millisecond-scale runtime overheads, making them unable to meet the strict sub-millisecond latency targets required by existing interactive microservices [14].
3. Serverless architectures can lead to increased reliance on a single cloud provider, raising concerns about vendor lock-in. This can be argued as a general cloud service issue and not specifically to FaaS. However often there is out-of-the-box integration with provider-specific services. For instance, AWS Lambda can natively be combined with Amazon SQS to use message queues as sources triggering function execution [31].

3 Scalability in Serverless Computing

Scalability, in the context of cloud computing, refers to the ability of a system to handle a growing amount of work or its potential to accommodate growth in demand. This typically includes the system's capacity to increase its performance proportionally with added resources, such as servers or storage. There are different notions of scalability as described in the examined literature, mainly the differentiation of horizontal and vertical scaling. The former meaning the addition of computing

nodes to the system and the latter the addition of computing power to a single node. [24] states that horizontal scaling is used in cloud computing environments [20]. Another differentiation of scalability definitions is proposed by [19], defined differently for both PaaS and SaaS respectively as "platform scalability is the ability of the execution platform to provide as many (additional) resources as needed (or explicitly requested) by an application" and "means that the application maintains its performance goals/SLAs even when its workload increases (up to a certain workload bound)"[19].

Serverless computing takes scalability to a new level by abstracting away the underlying infrastructure, allowing developers to focus solely on their application code. This abstraction eliminates the need for manual intervention in resource provisioning and load balancing, as the serverless platform automatically scales the application in response to demand. The automatic scaling characteristic of serverless computing is largely enabled by FaaS, as described in the Section 2. When a function is triggered by an event (e.g., an http request), the serverless platform automatically allocates the necessary resources to run the function. If the number of events increases, more instances of the function are automatically run in parallel.

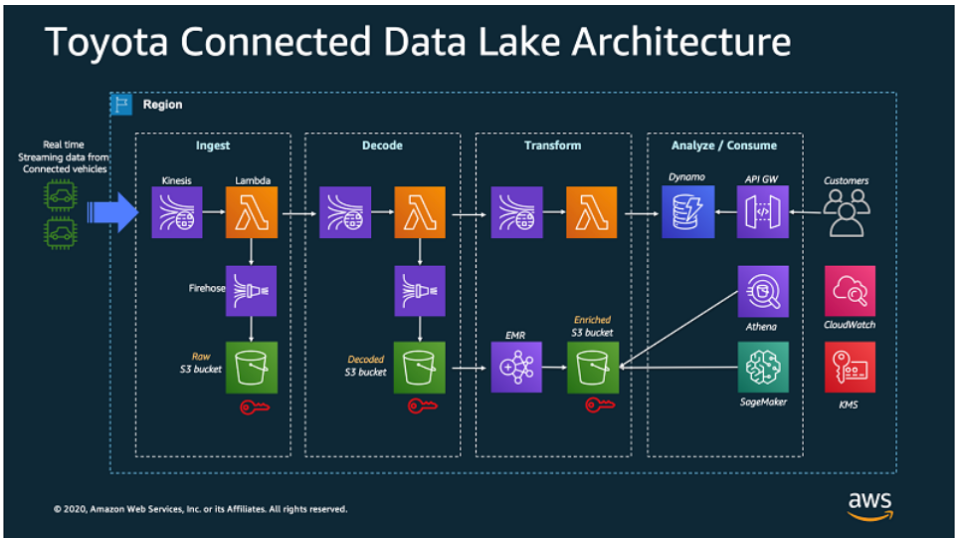


Figure 2: Case study example of Toyota Connected Serverless Architecture [18]

An example of scalability in serverless architectures comes from Toyota Connected. Toyota Connected is a subsidiary of Toyota, and it provides a range of technology services, including connected platforms, big data, mobility services, and other automotive-related services. They decided to build a platform using serverless architecture, specifically AWS Lambda, Amazon Kinesis Data Streams, and Amazon S3 [18]. Through this serverless architecture, Toyota Connected was able to scale up to 18 times its usual traffic volume, demonstrating the scalability of serverless. It was handling 18 billion transactions per month running through the platform. Additionally it also reduced data processing times by 97.5 percent, representing a significant increase in efficiency and a reduction in operational burden [18].

3.1 FaaS Execution at Scale

There are two main approaches to FaaS Execution at Scale. One is the traditional Container-Based Execution, the other is an emerging hybrid approach of Container and Virtual Machine.

Container-based execution is arguably the most popular approach in the FaaS landscape. In this model, each function runs inside a container, a lightweight standalone executable package, including the function code, runtime, system tools, and libraries. Services like Azure Functions use this approach. The container-based model's primary strength is that containers are lightweight compared to virtual machines, they can be started and stopped quickly, making them suitable for short execution time for FaaS workloads. Moreover, containers are isolated from each other, improving the security and reliability of functions. However, the container-based execution model also has some drawbacks. The cold start time can sometimes be longer than desirable. Furthermore, this model requires careful

management of container resources to prevent functions from exhausting their allocated resources [25].

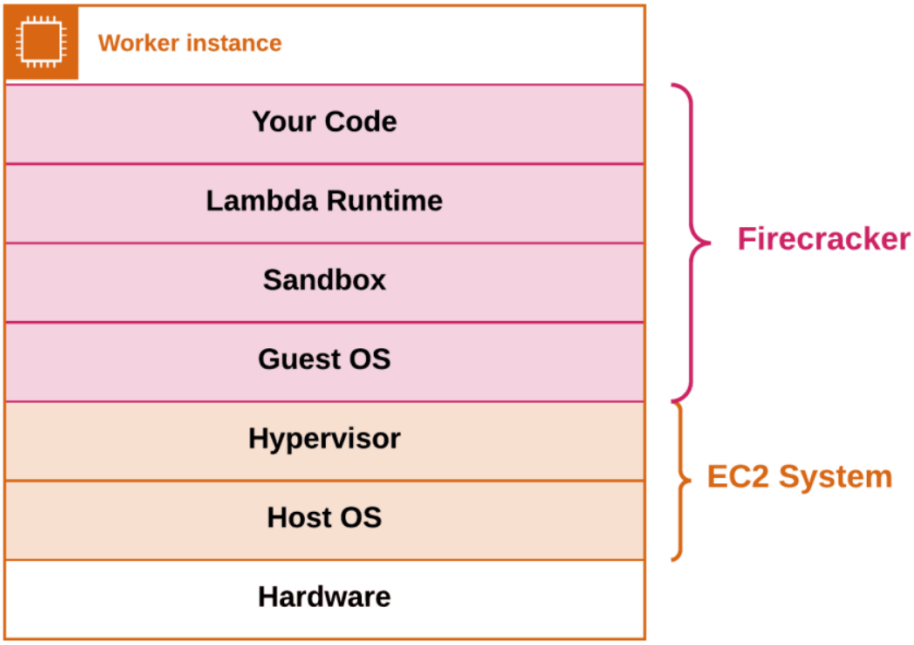


Figure 3: Architecture of Amazon Lambda Back-end [9]

The hybird method can be seen with Amazon Firecracker as seen in Figure 3 , which provides a unique approach to FaaS execution that combines elements of both VM-based and container-based execution. In the context of AWS Lambda, a FaaS offering by Amazon Web Services, Firecracker is used to provide a secure, multi-tenant isolation boundary. When a function is invoked, AWS Lambda uses Firecracker to create a microVM for that function. This microVM provides a level of isolation that is similar to what a traditional VM provides, but with the speed and resource efficiency that is closer to what containers provide. This allows AWS Lambda to securely run functions from multiple customers on the same physical machine without them being able to interfere with each other. Firecracker's microVMs can start up in as little as 125 ms and consume a fraction of the resources compared to traditional VMs [2].

3.2 Challenges with scaling in serverless computing

1. A key differentiator of serverless is the ability to scale to zero, or not charging customers for idle time. Scaling to zero, however, leads to the problem of cold starts, and paying the penalty of getting serverless code ready to run. The FaaS platform needs to allocate resources for the function, load the runtime, and then start the function, which can result in a significant delay. Techniques to minimize the cold start problem while still scaling to zero are critical [4].
2. Another challenge is that while serverless platforms handle many aspects of automatic scaling, they do not completely remove the need to consider scalability in the application design. For example, a serverless application that interacts with a traditional database may become a bottleneck if the database cannot scale as quickly or efficiently as the serverless functions.

4 Resiliency in Serverless Computing

The term resiliency is usually used to describe applications that are resistant to failure. A resilient system should be stable under unexpected changes to infrastructure or data [15].

On the one hand, Serverless architectures have some benefits with regards to resiliency. For one, the user can concern themselves less with damage and/or issues regarding the infrastructure, as many issues and/or damages will be handled by the cloud provider. Most cloud providers have automatic tests to check for any errors with hardware, and settings can be configured to restore and/or rerun any failed processes. This is dubbed 'retry based fault tolerance'.

Furthermore, a serverless architecture can also aid with communication between services. Cloud providers often offer infrastructure services such as queues to help implement robust communication patterns. Though they are not infallible, these services simplify the problem for developers, and are usually less likely to fail than custom code.

However, as mentioned in [17] , the reliability guarantees of the infrastructure provided by serverless computing (i.e. 'retry based fault tolerance') is often insufficient for applications requiring stronger notions of resilience such as in autonomous vehicles or analysis of live sensor feeds. It is especially an issue for applications that modify a shared state, as failures can cause partial updates that introduce faulty or outdated data into the state. Due to these issues, most reviews of Serverless Computing's state of the art consider resiliency to be insufficient in many cases, and it has been considered an open challenge [21].

4.1 Log-Based Runtimes

Recently, there have been many new attempts to address the challenge and improve resiliency of applications utilising serverless architectures. One attempt to improve the fault tolerance of serverless computing was made in [32]. This paper addresses the struggle to achieve a consistent and fault-tolerant critical state for stateful applications. An example to illustrate this problem is given in [13]: Suppose there is a travel reservation app built with serverless functions. When processing a reservation, the application might call a function to book a hotel as well as a function to book a flight, but if the functions fail it could result in an inconsistent state. The paper provides a solution to this by introducing a runtime for composing stateful serverless functions named Beldi. It extends a prior log-based approach, called Olive [26], with new data structures, algorithms, protocols, and garbage collection. Thus, the previous fault-tolerance approach is extended to stateless serverless functions, resulting in improved resiliency. The authors evaluated their approach by porting three applications to their runtime: a movie review service, a travel reservation application, and a social media site. Their results showed an increased, but still reasonable, latency (around 2-3x higher median response time when compared with the baseline, depending on the number of requests per second).

This approach was improved a year later in [13]. The authors implemented a new serverless runtime which implemented distributed shared logs by use of a LogBook API. Evaluations performed by the authors resulted in a performance improvement of up to 4.7x when compared to Beldi.

4.2 Fault-Tolerance Shims

As well as this, there are studies that aim to improve resilience for serverless computing in other ways, such as *A Fault-Tolerance Shim for Serverless Computing* [29]. In this paper, the authors introduce a low-overhead shim for serverless computing that enforces a read atomic isolation guarantee. This is a notion introduced in [3], where the authors state that "a system provides Read Atomic isolation (RA) if it prevents fractured reads anomalies and also prevents transactions from reading uncommitted, aborted, or intermediate data". In order to facilitate the design of this shim, the authors developed new protocols to guarantee read atomic isolation for shared storage, as well as a garbage collection scheme for these protocols to reduce the storage overhead.

The Master's Thesis by Saurav Chhatrapati, *Towards Achieving Stronger Isolation in Serverless Computing*[8], builds on this work. Here, the author suggests that Read Atomic Isolation does not prevent all consistency anomalies, and still allows some anomalies such as Lost Updates, Missing Dependencies, and Predicate-Many-Preceders, which can make it difficult for developers to reason about their applications. The author introduces a new low-overhead shim for serverless computing that enforces a stronger notion of consistency, Snapshot isolation, which guarantees Read Atomic Isolation and extends this by also preventing the aforementioned anomalies.

5 Designing Scalable and Resilient Serverless Applications

5.1 Designing Scalable Applications

As discussed in Section 3, Cloud platforms can scale Serverless applications for developers automatically, so there is less need for the developer to engineer a scalable design. However, it may still be necessary to consider the application design if the application interacts with other systems, such as external databases and APIs.

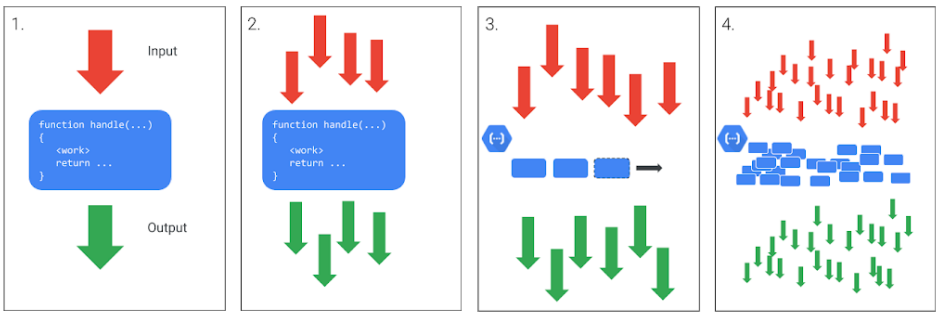


Figure 4: Scaling Patterns [1]

The Google Cloud blog[1] explains how FaaS scales using Figure 4. In 1, the most basic diagram, a single function takes an input and gives an output. In 2, the function is extended by taking multiple inputs and returning mulitple outputs. In 3, the serverless platform scales this horizontally, by providing multiple instances of the function(s). Finally, the fourth diagram visualises a highly scaled application, that may be bottle-necked by external systems.

Listed are some of the techniques that developers can use to solve the resulting problems, and ensure their serverless applications scale well:

1. Set Connection-Limits
2. Limit Rate of Work
3. Rate-Limit HTTP Requests
4. Process work in Batches
5. Use Managed Services
6. Perform Monitoring

5.2 Designing Resilient Applications

In order for a system to be resilient, it should be designed such that any unit/component can fail or be temporarily inaccessible without causing the entire system to fail. Additionally, components of the application should be able to resume after downtime with all data being preserved.

Even when using infrastructure services from a cloud provider, some consideration should be given to the communication pattern between services. For Instance, Depending on the use-case it may be preferable to use a Direct (Circuit Breaker) Communication pattern, or an Asynchronous Communication Pattern (as pictured in Figure 5. In case of an Asynchronous Communication Pattern, the developer will still need to account for downtime in a queue if one is being used, so the design would ordinarily include some form of data storage to be used in this case.

As in designing applications for scalability, there is a series of best practices that can be followed by developers to ensure good resiliency:

1. Modularity (Break the application into decoupled subprocesses)
2. Meaningful error handling and logging/monitoring
3. Design for idempotency, downtime, and high-throughput peaks
4. Build resiliency into workloads and communication patterns

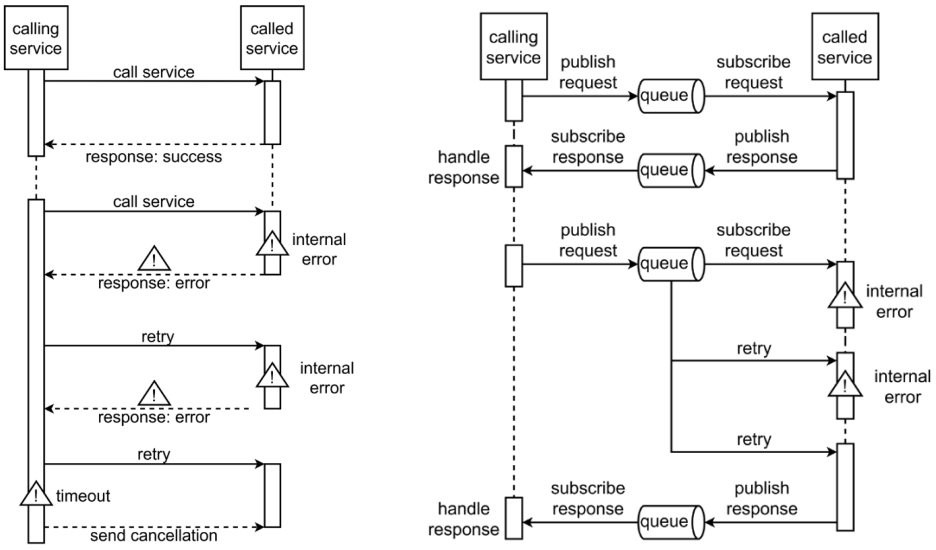


Figure 5: Direct Communication Pattern (Left) and Asynchronous Communication Pattern (Right) [15]

6 Discussion

6.1 Summary of Current Research

From the research we investigated, we formed a few main judgments. One of these is that although there are challenges to Scalability and Resiliency in Serverless Architectures, most users are satisfied with the services and benefits that cloud platforms provide in these areas, for example the automatic horizontal scaling of stateless serverless functions. Furthermore, by following best practices as outlined in Section 5, the majority of developers can solve most problems they encounter.

This is corroborated by [30], where authors analysed several hundred questions from Stack Overflow and categorised them into relevant groups. The paper did not find scalability or resiliency to be a common concern among developers.

We believe that scalability and resiliency become more significant problems in more specific use-case scenarios, such as when working with stateful serverless applications or functions. In these cases we have seen additional research that aims to solve these problems (at the cost of additional latency) such as log-based runtimes and fault-tolerance shims as described in Section 4. One compelling point of discussion is whether or not serverless architectures should be used for stateful architectures at all, or whether the drawbacks outweight the benefits. We consider this in the following section.

6.2 When to use Serverless (Stateful vs Stateless)

Common reasons not to use Serverless Architectures include Vendor Lock-In, Latency concerns and unique security risks. We have mentioned some of these in Section 2. Another common critique of Serverless computing is that it is not very compatible with Stateful applications or functions.

Indeed, there are some drawbacks to developing stateful applications using serverless architectures. Firstly, there is limited scalability - the automatic horizontal scaling that is done with stateless serverless functions does not translate directly to stateful functions as there may be a need for synchronising a share state accross multiple instances of functions. Secondly, as explored in Section 4, ‘retry-based fault tolerance’ is often inadequate in stateful applications, creating a need for alternative solutions. Furthermore, the additional complexity of the application design due to the need for synchronisation of state might result in a more difficult development process, as well as higher latency and cost.

However, as we have seen in Section 4, there are ways to solve the problems introduced by Stateful Serverless Computing, meaning it can be an effective approach in some use-cases. Moreover, there exist other techniques for Stateful Serverless Computing that we have not explored in depth in this literature review [28, 16, 5, 6].

In closing, we believe that the decision of whether to use Serverless computing for developing stateful applications (or in general) depends highly on the specific use-case, and the requirements and constraints associated with it. If, for example the serverless approach would introduce a large amount of excess complexity, such as in the case of a machine learning application, it may be preferable to use a traditional server based approach.

6.3 Future Research Possibilities

In future, researchers can continue looking into different techniques to increase the ease of Stateful Serverless application development, such as novel approaches for managing the shared state. Researchers could also improve scalability and resiliency by examining the strategies used by current cloud platforms and considering improvements that could be made. For example, researchers could look at different methods of resource allocation in the case of scalability, or investigate fault tolerance by use of chaos engineering and fault injection. A final area that could be analysed in higher depth are the emerging hybrid approaches to FaaS Execution at Scale, as discussed in Section 3.

7 Conclusion

In conclusion, We answered how serverless architectures impact application scalability and resiliency by investigating existing studies based on existing applications, reviews of state of the art, and other recent research in these topics. We established that serverless computing provides numerous benefits to both scalability and resiliency, primarily in the form of automatic horizontal scaling of stateless serverless functions and retry-based fault tolerance respectively. Moreover, we investigated the drawbacks on scalability and resiliency in stateful applications, and discussed whether or not serverless architectures should be used for them. Finally, We answered how to design serverless applications that are scalable and resilient by describing some of the more relevant industry practices. An implication of our study is that some of the main weak points of serverless architectures are in developing stateful applications. We suggest that future research could continue to explore this area.

References

[1] URL: https://cloud.google.com/blog/products/serverless/6-strategies-for-scaling-your-serverless-applications.

[2] Alexandru Agache et al. "Firecracker: Lightweight Virtualization for Serverless Applications." In: *NSDI*. Vol. 20. 2020, pp. 419–434.

[3] Peter D. Bailis et al. "Scalable Atomic Visibility with RAMP Transactions". In: *ACM Transactions on Database Systems (TODS)* 41 (2014), pp. 1–45.

[4] Ioana Baldini et al. "Serverless computing: Current trends and open problems". In: *Research advances in cloud computing* (2017), pp. 1–20.

[5] Daniel Barcelona-Pons et al. "On the FaaS Track: Building Stateful Distributed Applications with Serverless Architectures". In: *Proceedings of the 20th International Middleware Conference*. Middleware '19. Davis, CA, USA: Association for Computing Machinery, 2019, pp. 41–54. ISBN: 9781450370097. DOI: 10.1145/3361525.3361535. URL: https://doi.org/10.1145/3361525.3361535.

[6] Daniel Barcelona-Pons et al. "Stateful Serverless Computing with Crucial". In: *ACM Trans. Softw. Eng. Methodol.* 31.3 (Mar. 2022). ISSN: 1049-331X. DOI: 10.1145/3490386. URL: https://doi.org/10.1145/3490386.

[7] Paul Castro et al. "The rise of serverless computing". In: *Communications of the ACM* 62.12 (2019), pp. 44–54.

[8] Saurav Chhatrapati. "Towards Achieving Stronger Isolation in Serverless Computing". MA thesis. EECS Department, University of California, Berkeley, May 2021. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-141.html.

[9] Damian. *Anatomy of AWS Lambda*. dev.to. [Online; accessed 1-June-2023]. 2019. URL: https://dev.to/sosnowski/anatomy-of-aws-lambda-1i1e.

[10] Geoffrey C Fox et al. "Status of serverless computing and function-as-a-service (faas) in industry and research". In: *arXiv preprint arXiv:1708.08028* (2017).

[11] Vânia Gonçalves and Pieter Ballon. "Adding value to the network: Mobile operators' experiments with Software-as-a-Service and Platform-as-a-Service models". In: *Telematics and Informatics* 28.1 (2011), pp. 12–21.

[12] Hassan B Hassan, Saman A Barakat, and Qusay I Sarhan. "Survey on serverless computing". In: *Journal of Cloud Computing* 10.1 (2021), pp. 1–29.

[13] Zhipeng Jia and Emmett Witchel. "Boki: Stateful Serverless Computing with Shared Logs". In: *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. SOSP '21. Virtual Event, Germany: Association for Computing Machinery, 2021, pp. 691–707. ISBN: 9781450387095. DOI: 10.1145/3477132.3483541. URL: https://doi.org/10.1145/3477132.3483541.

[14] Zhipeng Jia and Emmett Witchel. "Nightcore: efficient and scalable serverless computing for latency-sensitive, interactive microservices". In: *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021, pp. 152–166.

[15] Benjamin Kettner and Frank Geisler. "Achieving Resiliency". In: *Pro serverless data handling with Microsoft azure: Architecting ETL and data driven applications in the cloud*. APRESS, 2022.

[16] Anurag Khandelwal et al. "Jiffy: Elastic Far-Memory for Stateful Serverless Analytics". In: *Proceedings of the Seventeenth European Conference on Computer Systems*. EuroSys '22. Rennes, France: Association for Computing Machinery, 2022, pp. 697–713. ISBN: 9781450391627. DOI: 10.1145/3492321.3527539. URL: https://doi.org/10.1145/3492321.3527539.

[17] Sameer G Kulkarni et al. "Living on the Edge: Serverless Computing and the Cost of Failure Resiliency". In: *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. 2019, pp. 1–6. DOI: 10.1109/LANMAN.2019.8846970.

[18] Sandeep Kulkarni and Shravanthi Denthumdas. *Enhancing customer safety by leveraging the scalable, secure, and cost-optimized Toyota Connected Data Lake*. https://aws.amazon.com/blogs/big-data/enhancing-customer-safety-by-leveraging-the-scalable-secure-and-cost-optimized-toyota-connected-data-lake/. Accessed: 2023-05-30. Amazon Web Services, Aug. 2020.

[19] Michael Kuperberg et al. *Defining and quantifying elasticity of resources in cloud computing and scalable platforms*. KIT, Fakultät für Informatik, 2011.

[20] Sebastian Lehrig, Hendrik Eikerling, and Steffen Becker. "Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics". In: *Proceedings of the 11th international ACM SIGSOFT conference on quality of software architectures*. 2015, pp. 83–92.

[21] Yongkang Li et al. "Serverless Computing: State-of-the-Art, Challenges and Opportunities". In: *IEEE Transactions on Services Computing* 16.2 (2023), pp. 1522–1539. DOI: 10.1109/TSC.2022.3166553.

[22] Sunilkumar S Manvi and Gopal Krishna Shyam. "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey". In: *Journal of network and computer applications* 41 (2014), pp. 424–440.

[23] Joao Menezes Carreira. "Practical and Scalable Serverless Computing". PhD thesis. EECS Department, University of California, Berkeley, Dec. 2021. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-238.html.

[24] Zoltán Micskei. "Dynamically Scalable Applications Cloud Environment". PhD thesis. Dissertation, Budapest University of Technology and Economics, Budapest ...

[25] Jungae Park, Hyunjune Kim, and Kyungyong Lee. "Evaluating Concurrent Executions of Multiple Function-as-a-Service Runtimes with MicroVM". In: *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. 2020, pp. 532–536. DOI: 10.1109/CLOUD49709.2020.00080.

[26] Srinath T. V. Setty et al. "Realizing the Fault-Tolerance Promise of Cloud Storage Using Locks with Intent". In: *USENIX Symposium on Operating Systems Design and Implementation*. 2016.

[27] Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. "Architectural implications of function-as-a-service computing". In: *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*. 2019, pp. 1063–1075.

[28] Simon Shillaker and Peter R. Pietzuch. "Faasm: Lightweight Isolation for Efficient Stateful Serverless Computing". In: *CoRR* abs/2002.09344 (2020). arXiv: 2002.09344. URL: https://arxiv.org/abs/2002.09344.

[29] Vikram Sreekanti et al. *A Fault-Tolerance Shim for Serverless Computing*. Mar. 2020.

[30] Jinfeng Wen et al. "An Empirical Study on Challenges of Application Development in Serverless Computing". In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Athens, Greece: Association for Computing Machinery, 2021, pp. 416–428. ISBN: 9781450385626. DOI: 10.1145/3468264.3468558. URL: https://doi.org/10.1145/3468264.3468558.

[31] Vladimir Yussupov et al. "FaaSten your decisions: A classification framework and technology review of function-as-a-Service platforms". In: *Journal of Systems and Software* 175 (2021), p. 110906.

[32] Haoran Zhang et al. "Fault-tolerant and transactional stateful serverless workflows". In: *USENIX Symposium on Operating Systems Design and Implementation*. 2020.

Addressing Challenges of Fog Computing Using Blockchain Technology

Laréb Fatima Ahmad
14118114
Computer Science
University of Amsterdam
fatima.ahmad@student.uva.nl

Vishal Kanteppa Mahesh
14740583
Computer Science
University of Amsterdam
vishal.mahesh@student.uva.nl

Aditya Srivastava
14288060
Computer Science
University of Amsterdam
aditya.srivastava@student.uva.nl

1 Introduction

The concept of fog computing has gained significant attention as a potential solution for enhancing how data is processed and analyzed. Rather than relying exclusively on distant servers like conventional cloud computing approaches do, fog computing moves computation capabilities closer to the data source. This shift makes it possible to quickly and efficiently draw meaningful insights from large amounts of data before taking necessary actions [1].

Although fog computing addresses challenges with cloud computing such as offering cloud services with lower latency and with ad hoc adaptations, it comes with its own challenges. For instance, it is integral to ensure secure connections between distributed fog nodes and between nodes and mobile devices, guarantee nodes can scale according to network traffic and have a rigid fault tolerance system in order to avoid possibly detrimental consequences of failures or significant loss of data [2], [3].

Blockchain's cutting-edge encryption algorithms and consensus mechanisms can be used in fog ecosystems and offer new ways of ensuring transparency, accountability, and resilience in environments including countless connected devices [4], [5]. Blockchain is a novel technology made to simplify interactions with multiple parties through distributing a shared database, also called a ledger. By sharing copies of the ledger across a computer network, alterations of it become near impossible. As such, blockchain can enhance security, reduce expenses, speed up processes, increase transparency of fog computing[6].

This study seeks to answer a crucial research question: *How can blockchain technology be effectively utilized to enhance the security and trustworthiness of fog computing systems?* Through literature reviews, this paper sheds light on these two advancing domains and discusses solutions to challenges with fog computing with the help of blockchain. It will consider how blockchain can enhance fog computing by contributing to data integrity, managing access control and increasing privacy and trustworthiness of data transactions in the fog.

The setup of the current paper is as following: section 2 and 3 describe fog computing and blockchain, section 4 introduces some challenges with fog computing and section 5 discusses how the many characteristics and many benefits of blockchain technology can be used to address the issues of fog computing. Section 6 concludes the paper, and section 7 considers future studies on the highlighted topic.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 Fog Computing

Fog Computing was introduced by CISCO in 2011 [7]. Data is accessed typically at the "edge" of the cloud. The fog originates from a wish to bring resources closer to the end-user by creating endpoints at the edges of the network, for instance executing processing in the fog for online gaming, where low latency is important [7].

Figure 1 shows an overview of the deployment of the fog. It is not separate from the cloud, but rather an extension of it, or an additional layer between the cloud and the end-user devices. While some processing is executed at the edge of the cloud, some are still sent to the cloud [1].

Architecturally, fog computing consists of three layers: the terminal layer, the fog layer and the cloud layer. The terminal layer is where the end-user devices lie such as smart phones, tablets, computers and other types of IoT devices such as smart home appliances [8]. Data from the terminal layer is transmitted to the edge of the network, to fog layer. This layer constitutes of numerous fog nodes, such as servers, access points and gateways in which some computing, storage and transmission happens [8]. The fog layer interacts with both the terminal layer and the cloud layer. This layer is the "cloud", which has the same capabilities as the fog layer but on a much larger scale and power level and more resources [8].

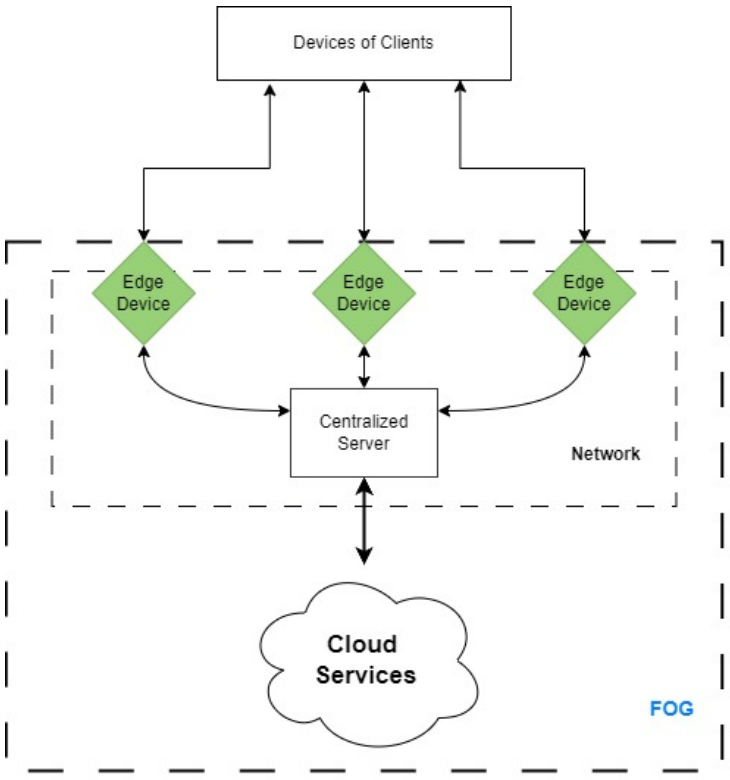


Figure 1: Fog Computing's deployment overview

What makes up the fog (and also the cloud) are resources for computing for data processing, secure network to transfer data, and storage of client data for the cloud [1]. The fog is characterized by having many devices deployed over large distances, such as on highways or spread across a city. In contrast to the cloud, the fog is decentralized and distributed [7], [9], [10]. Additionally, as fog nodes are used in various environments, heterogeneity is another important factor of fog nodes, in terms of adapting to their specific use cases, environments and devices in which they are deployed [1], [11]. Fog nodes can be installed in devices that are static or moving, such as a car. A consequence of fog's localization abilities and heterogeneity is context awareness, contributing to the fog's ad hoc abilities.

Fog computing primarily deals with sharing devices capable of running applications locally and communicating directly with each other. This involves performing cloud-like functions, such as data computation, data query, data storage, and data sharing services [9], [11], on decentralized devices near the points of data creation. This allows for data to be processed closer to where it originates, without necessarily having to send it back to a centralized server or the cloud [2]. This improves efficiency and reduces latency and resource consumption through less raw data being transmitted over large networks, and consequently reduces processing costs [1], [3]. Fog computing is thus especially suitable for use cases such as real-time control and monitoring and handling large amounts of Internet of Things (IoT) device data. [2], [9].

Another important feature of the fog is support for mobility by communicating with mobile devices directly, and with a wide range of devices [3], [7]. The fog uses the LISP protocol, which contributes to effective distribution over wide areas and connection of many fog nodes. LISP is a routing architecture created by CISCO inc, which splits device location and device identity. Separating the two improves scalability and routing [12].

2.1 Cloud Computing

The cloud allows for more computational power and processing of large volumes of data, in one centralized place. Software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) are the three main services which cloud computing is utilized for. Examples include Amazon Web Services, Microsoft Azure, and Google Cloud Platform, all cloud services provided by large technology corporations which offer one of more of the three services [1], [10].

Alternatively to fog computing, cloud computing processes data remotely on shared resources accessed in remote servers over wide area networks such as the Internet or other centralized data centers [11]. This data is usually stored far away from the user, thus may result in higher latency when accessed as compared to data in the fog. As the cloud is online and relies on access to the Internet, it is also susceptible to the same security issues as the Internet. Deployment in the fog mitigates this issue to some degree as the data travels shorter distances and can be accessed offline [1], [10].

A precursor to the fog is "mini clouds" or "edge clouds", which are smaller clouds placed at the edges of the network that work as entry points for IoT devices and in order to confine network traffic within its bounds and bring resources closer to the end device, much like the purpose of the fog today, as Vaquero and Rodero-Merino describe in their 2014 paper [9] and Chang et al discuss in their paper from the same year [13].

3 Blockchain

A blockchain serves as a digital ledger recording various types of transactions and interactions occurring within its ecosystem. Each entry, called a block, holds numerous transactions, along with a timestamp, a unique identifier representing the previous block (known as a parent hash), and a randomly generated number value (a nonce) necessary for verifying the hash and that can only be used once [4]. Before a block gets added to the blockchain, these components undergo validation processes agreed upon by participating nodes in the network [4], [14]. Every blockchain maintains multiple copies of its entire history, stored on different nodes throughout the distributed system. To guarantee consistency among nodes, a peer-to-peer (P2P) validation mechanism helps nodes compare their local versions of the blockchain with each other [4]. Figure 2 shows an exmaple of a blockchain, and what a block consists of, where "TX" is a transaction.

The use of cryptographic techniques, such as hash functions, provides additional safety measures by making it harder for potential cybercrime attacks [6]. Changes made to transactions or files will also change the corresponding hash values in subsequent blocks, making it easy to detect tampering [14]. Timestamp recording, in addition to the hash-values, lowers the risk of deception [6]. The distributed nature of blockchain works as a safety mechanism in case of network disruptions as it lessens the potential of single-point of failures. Another feature of the non-centralized structure is the lack of third-parties involved, which again increases trustworthiness and user privacy by lowering the risk of data breaches or access to user data by intermediaries [4]. Furthermore, once written, information becomes essentially immutable, which makes blockchain a sturdy and secure technology that can

further promote data integrity [6], [14]. Overall, blockchain technology bring advanced security to digital record-keeping by establishing confidence in the accuracy and integrity of transactions.

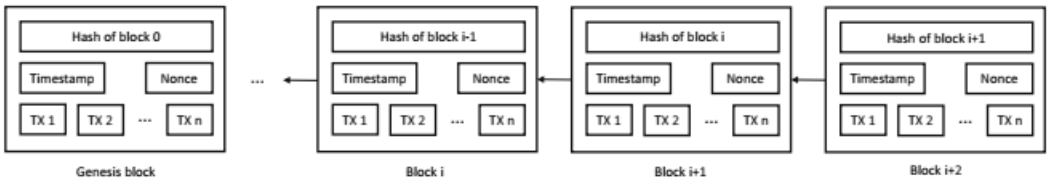


Figure 2: Example of a blockchain [4]

Section 5 discusses how the many inherent abilities of blockchain technology can provide many benefits to fog computing.

3.1 Bitcoin

In 2008, the crypto currency Bitcoin was introduced by a person or several people under the name "Satoshi Nakamoto" [15]. In 2009, Bitcoin was released as open source and has since become the most popular usage of blockchains, specifically transactions in Bitcoin and the validation of these [15].

Validating transactions through P2P reviews is the principle of Bitcoin mining. All computers on a blockchain network has access to the exact same files. Bitcoin mining is essentially a race to validate the hash of a newly added block in the chain for a reward in Bitcoins [14], [15]. Also called Token Economics, this ensures that a transaction can only be sent if the sender has sufficient funds, and the node selected to verify the transaction is rewarded in bitcoins [16].

4 Challenges with Fog Computing

This section includes an extensive discussion of challenges with fog computing. Specifically, the context of the following discussion are challenges with privacy, security, scalability, mobility, fault tolerance and reliability are addressed, why these challenges occur in fog ecosystems and the requirements for mitigating these challenges.

4.1 Privacy and Security

Due to its scattered nature and the availability of linked devices, fog computing presents new security and privacy problems. Fog nodes and devices must create encrypted channels to secure data transmission from unauthorized access. Secure communication is essential as fog computing involves processing sensitive data at the edge, privacy protection becomes particularly critical. User privacy can be protected by using privacy-preserving methods such secure data aggregation and data anonymization. While data anonymization [17] techniques protect personal information by de-identifying or concealing data, secure data aggregation [18] integrates data from numerous sources while protecting individual privacy. For identifying and reducing security risks in fog computing, continuous security monitoring, intrusion detection systems (IDS) [19], and anomaly detection procedures are crucial. The security posture of the foggy environment is maintained through proactive monitoring and prompt response to security issues [20]. While anomaly detection techniques enable the detection of unexpected patterns or behaviors that may indicate a security breach, IDS can spot possible threats and abnormalities.

4.2 Data Integrity

Data integrity is another critical aspect in fog computing. Fog nodes should verify the accuracy and integrity of the data they receive from sensors or other devices before transmitting or processing it. This verification helps ensure that compromised or tampered data does not propagate throughout the fog network, maintaining the integrity of the overall system [21]. Role-based access control and fine-grained access control are two access control strategies that are crucial for preventing unauthorized

access to sensitive resources and data. Since fog computing includes processing sensitive data at the edge, privacy protection is particularly crucial [22]. User privacy can be preserved via methods like safe data aggregation and data anonymization. Detecting and reducing security risks requires constant security monitoring, intrusion detection systems, and anomaly detection procedures [23].

4.3 Scalability and Mobility

Due to its dynamic nature, where devices and services can enter or exit the network at any time, fog computing presents substantial issues in terms of scalability and mobility [24]. It is a difficult task to guarantee uninterrupted service delivery while managing dynamic connectivity and ensuring seamless handover [25].

To deal with these issues, mechanisms like Mobile IP and seamless roaming have been created. As the number of devices and services rises, load balancing becomes more and more important [26]. To avoid performance bottlenecks and maximize system effectiveness, the workload must be distributed among fog nodes efficiently while taking into account their processing power and resource availability. It is a difficult task to manage network latency in a dispersed system with varied network conditions and multiple devices. Cutting down on latency, increasing responsiveness, and boosting the user experience all depend heavily on strategies like edge caching, adaptive routing, and real-time traffic management [27]. To efficiently manage the ever-increasing scale of fog computing environments, scalable resource management techniques, decentralized decision-making algorithms, and effective resource provisioning systems are also essential.

4.4 Fault Tolerance and Reliability

Fog computing systems must be fault tolerant and reliable in order to handle failures, disturbances, and unstable network conditions successfully [28]. Redundancy and fault-tolerant systems must be put in place for reliable service delivery. Even in the event of failures or disruptions, the use of redundant fog nodes, backup assets, and replication techniques helps ensure service availability. Additionally, it's crucial to control network congestion, which might happen as a result of a large number of devices and data traffic [29].

Fog computing systems can reduce network congestion and guarantee reliable data transfer by applying congestion-aware routing, traffic shaping, and Quality of Service (QoS) management techniques. Due to the variety of resources and devices, detecting errors and failures in a distributed fog environment can be difficult. To sustain system reliability, it is essential to integrate fault detection systems, proactive health monitoring, and efficient fault recovery processes [30]. Fog computing systems can achieve continuous operation and provide dependable services to end users by utilizing fault-tolerant techniques and backup options.

5 Addressing Challenges of Fog Computing with Blockchain

A possible paradigm for enabling effective and decentralized computation at the network's edge is fog computing. However, it also presents a number of difficulties, particularly in terms of privacy and security. To achieve effective fog computing and overcome its challenges, establishing robust security features is imperative. The incorporation of blockchain technology holds a lot of promise for overcoming these difficulties. Blockchain, which is renowned for being decentralized[31], open, and impervious to manipulation, can offer ground-breaking solutions to improve security, safeguard personal information, and build trust in fog computing environments. It can create secure communication channels, maintain data integrity through tamper-proofed records, apply fine-grained access controls, preserve privacy via encrypted data sharing, develop trust and reputation models, and establish consensus protocols among fog nodes[32]. Furthermore, organizations seeking to adopt fog computing can leverage these strengthened capabilities to deliver high-quality services, reduce costs, and mitigate risks in today's dynamic business landscape.

5.1 Secure Communication

Secure communication is a crucial component of fog computing, especially when dealing with sensitive data, such as data concerning private health information. To address this concern, utilizing

blockchain as a decentralized framework establishes a highly resilient means for communication. Blockchain uses advanced cryptographic protocols and consensus methods to guarantee immutable records and prevent malicious intrusions [6], [33]. For instance, private keys can be generated and securely distributed via blockchain, allowing fog nodes to safely communicate without risking exposure to single points of failure. Blockchain's immutability feature guarantees unchanging secret key, reducing potential of attacks. Additionally, if a node were to be compromised, other nodes in the network could continue functioning without incurring further damage as they are distributed.

Hash values stored onto the blockchain create irrefutable cryptographic proofs that specific data was transmitted at a given point in time [14], [15]. Permanently recorded transmission records provide accountability and deter against false claims or misinformation spread through fog networks. Additionally, a secure connection enabled by blockchain's solid structure protects sensitive data from being exposed to potential threats outside of the fog network. This added layer of protection can increase confidence and trust among participants.

These features create a robust foundation for secure data exchange within the fog environment, solidifies the integrity of data and prevents unwanted access.

5.2 Data Integrity

Ensuring the authenticity and validity of sensor data becomes critical when implementing fog computing systems. Utilizing blockchain as a validation tool provides multiple advantages for achieving these objectives. One significant factor is its immutability feature, where once recorded, data cannot be tampered with [4]. Thus, any changes made to the data must go through a validated process involving multiple parties, ensuring data consistency and accuracy over time [4], [14]. By employing blockchain technologies, fog nodes can thoroughly examine and affirm the accuracy of information received from connected devices without fear of compromise. Any attempt to modify data on the distributed ledger can be instantly detected and flagged by the nodes. This creates a robust mechanism against fraudulent modifications and increases data trustworthiness.

Transparent record-keeping is another key attribute offered by blockchain, allowing stakeholders to monitor transaction activity in real-time [6]. By having complete visibility into all aspects of a fog network, stakeholders and decision makers alike can quickly identify anomalous behavior, monitor and confirm data transactions and ensure accountability within the fog network. Additionally, disputes concerning authenticity, authorship, or provenance can be resolved with high confidence since every node shares identical copies of the shared history [5], [34].

5.3 Access Control

Systems for managing identities based on blockchains in fog computing settings provide decentralized and secure access control. By using blockchains to manage identities, one can ensure that each device has a distinct digital signature that can be verified across the entire system. This makes it possible to control access to data and resources in a decentralized manner [33].

Smart contracts can be used to manage access rights, giving users granular control and lowering the possibility of illegal access to private information. Smart contracts are self-executing pieces of code that automatically enforce rules and terms as defined by their developers. They can be used to manage access rights to different resources and services within the fog network [35]. For example, a user might create a smart contract that specifies who has permission to view certain documents, under what conditions those permissions may be revoked, and so forth. These contracts are stored on the blockchain, making them immutable and transparent, which helps further increase security and trustworthiness [36], [37].

5.4 Privacy Protection

Blockchain technology makes it possible for fog computing to use privacy-preserving processes like private smart contracts and zero-knowledge proofs [35]. These methods protect user privacy and the confidentiality of their data by enabling sensitive data to be securely processed and authenticated on the blockchain without disclosing the underlying data [33]. For instance, instead of revealing customer names or addresses when conducting transactions between peers, encrypted tokens which represent this data are used.

As data is spread over numerous nodes instead of concentrating it in centralized repositories, attack surfaces decrease. Furthermore, as blockchain relies on a combination of cryptography and peer validation, dependencies on and data exposure to external sources and third-parties is reduced. Additionally, trustless architectures eliminate reliance on intermediaries, promoting greater individual control over personal info, such as smart contracts and predefined consensus mechanisms. Smart contracts also allow for specifications of detailed rules on access control [36], [37].

5.5 Trust and Reputation Management

Blockchain's decentralization and transparency enable reputation management inside the fog computing ecosystem. Stakeholders can build confidence and evaluate the dependability of fog nodes, devices, and services by documenting transactions, interactions, and performance data on the blockchain . Supply chain tracking helps ensure components are genuine and meet specified quality criteria [34]. This visibility and traceability enhances consumer trust as they can check whether a anything has been tampered with along its journey. As a result, dangers brought on by hostile or untrustworthy entities are reduced and a trustworthy atmosphere is fostered.

Additionally, blockchain offers opportunities for distributed storage across a network instead of using cloud services to host data on fog devices. This also makes it more challenging to interfere with transactions and ensures data integrity, thus enhancing privacy [14], [22].

5.6 Consensus Algorithms and Distributed Ledger

Blockchain technology can allow for communities to organize and reach consensus on desired features for next-generation fog devices. Token economics can be used, which enable effective alignment of interests among all stakeholders and consumers [5], [16]. This contributes to mutually beneficial outcomes. As a result, customers will experience advancements which address their specific needs, something which can lead to increased loyalty and trustworthiness.

Consensus algorithms are used in blockchain technology to ensure agreement and confirmation of transactions within the fog network. The integrity and immutability of data recorded on the distributed ledger are guaranteed by consensus techniques like Proof of Work (PoW) and Proof of Stake (PoS) [15], [38]. Fog computing systems can create a trustworthy, decentralized environment where fog nodes can agree on the legitimacy of transactions by using consensus methods. The distributed ledger offers a clear and impenetrable record of every transaction, enabling auditing, traceability, and protecting the system's integrity [6].

It is crucial to carefully design and evaluate the implementation of blockchain technology in fog computing, considering factors such as scalability, latency, and energy efficiency. By taking these considerations into account, fog computing can fully leverage the benefits of blockchain to address security and privacy challenges effectively.

6 Conclusions

Fog computing works as an extra layer between the cloud and the end-user. The fog is a closer to the user than the cloud, and consists of many of the same resources as it such as computation power, storage capabilities and network for transmitting data between the users' devices, the fog and the cloud. It is decentralized, as opposed to the centralized cloud, and spread over large geographical areas. As such, end-user devices can access the fog instead of the cloud for faster responses, such as low-latency in online gaming.

Blockchain is a robust technology which works as a digital ledger for storing transactions between nodes in a network. Additionally, it adheres to a distributed structure where every node in its network keeps an exact copy of the ledger. Blockchain technology has many inherent perks, such as immutable and historic record-keeping which increase the integrity of the data being stored, and raises flags immediately is data is tampered with.

Fog computing and blockchain are two powerful technologies that have shown great potential individually. When combined properly, however, these technology can blockchain can meet fog computing where it lacks, and possibly build an even stronger, trustful, efficient, cost-effective, secure, safe, private, and reliable fog computing framework.

The use of blockchain as a decentralized framework in fog computing ensures secure communication. Cryptographic protocols and consensus techniques used in blockchain provide an impenetrable channel of communication, such as with the use of private keys. Immutable storage of hash values creates indisputable proof of data transmission, providing traceability. The combination of blockchain and fog computing promotes reliability, security, and trustworthiness of sensitive data exchange. Data cannot be tampered with once the block containing it is added to the ledger. Any changes must go through a communal validation system, and all activities will be recorded by all nodes in the network. Fog nodes can use this mechanism to enhance transparency and security among stakeholders. Fog computing thrives under a decentralized setting; therefore, applying decentralized identity and access control solutions befitting the fog's structure is highly useful, such as with blockchain. Using smart contracts that run on top of blockchain technology provides granular access rights to various resources and services within the fog network. Smart contracts in blockchains are automatically executed access management protocols.

Blockchain can significantly increase privacy protection and reduce risks associated with data breaches by allowing secure processing and authentication of sensitive data without exposing it. Coupling fog computing with zero-knowledge proofs also increases trust between parties through encrypted representations of personally identifiable details. In addition to the decentralized structures, the lack of third-parties involved, timestamp recording, supply-chain tracking and encrypted distributed storaging provided by blockchain contributes to enhance the trust and and reputation of fog computing for the stakeholders involved. Blockchain's consensus algorithm also contributes to transparency among stakeholders and end-users by providing full insight into all actions taken with the use of immutable record-keeping.

7 Future Research

The combination of fog computing with blockchain technology is still in its infancy and requires more research [32], [39]. The following section discusses some of the areas of the technology which requires more attention and research.

7.1 Lack of Standardization

A significant challenge pertains to differences in protocol standards. Fog computing and blockchain has evolved at varying rates and adopted distinct rules. To combine them successfully, standardization efforts would need to catch up. If not addressed early enough, differing frameworks may disrupt functioning.

7.2 Scalability Limitations

For fog computing to handle millions of IoT devices, scaling remains essential. Conventional blockchain designs tend to face difficulties maintaining their original speeds because of high volumes [4], [6], [37]. This problem might persist after combining fog and blockchain technology unless addressed appropriately as blockchain does not necessarily improve the scalability issues of fog computing. Therefore, research is required to optimize their integration to prevent bottlenecks and slowdowns in decision-making and information sharing.

7.3 Energy Efficiency Concerns

Current economic models incentivize individuals to invest heavily in infrastructure, including specialized hardware and high-end cooling systems. While these additions improve overall efficiency, they increase energy needs dramatically [4], [15]. PoW is a popular consensus mechanism which results in high levels of energy consumption [15], [37], [38]. With more users and transactions on the network, the energy usage will rise proportionally. Lastly, as the size of the blocks increases and the number of transactions grows, each block becomes heavier, leading to longer synchronization times and increased bandwidth consumption [6], [15]. This results in higher resource costs and greater energy demand.

References

[1] V. Kumar, A. A. Laghari, S. Karim, M. Shakir, and A. Anwar Brohi, "Comparison of fog computing and cloud computing," *International Journal of Mathematical Sciences and Computing*, vol. 5, no. 1, pp. 31–41, Jan. 2019. DOI: 10.5815/ijmsc.2019.01.03. [Online]. Available: http://www.mecs-press.org/ijmsc/ijmsc-v5-n1/IJMSC-V5-N1-3.pdf.

[2] S. Chen, T. Zhang, and W. Shi, "Fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 4–6, 2017. DOI: 10.1109/MIC.2017.39. [Online]. Available: https://ieeexplore.ieee.org/document/7867739.

[3] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, 2015, pp. 73–78. DOI: 10.1109/HotWeb.2015.22. [Online]. Available: https://ieeexplore.ieee.org/document/7372286/.

[4] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Business and Information Systems Engineering*, vol. 59, no. 3, pp. 183–187, Mar. 2017. DOI: 10.1007/s12599-017-0467-3. [Online]. Available: https://link-springer-com.vu-nl.idm.oclc.org/article/10.1007/s12599-017-0467-3#citeas.

[5] J. Y. Lee, "A decentralized token economy: How blockchain and cryptocurrency can revolutionize business," *Business Horizons*, vol. 62, no. 6, pp. 773–784, 2019, Digital Transformation Disruption, ISSN: 0007-6813. DOI: https://doi.org/10.1016/j.bushor.2019.08.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007681319301156.

[6] U. Bodkhe, S. Tanwar, K. Parekh, *et al.*, "Blockchain for industry 4.0: A comprehensive review," *IEEE Access*, vol. 8, pp. 79 764–79 800, 2020. DOI: 10.1109/ACCESS.2020.2988579. [Online]. Available: https://ieeexplore-ieee-org.vu-nl.idm.oclc.org/abstract/document/9069885.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," ser. MCC '12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 13–16, ISBN: 9781450315197. DOI: 10.1145/2342509.2342513. [Online]. Available: https://doi.org/10.1145/2342509.2342513.

[8] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017, ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.09.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517302953.

[9] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014, ISSN: 0146-4833. DOI: 10.1145/2677046.2677052. [Online]. Available: https://dl.acm.org/doi/10.1145/2677046.2677052.

[10] H. Rashid Abdulqadir, S. R. M. Zeebaree, H. M. Shukur, *et al.*, "A Study of Moving from Cloud Computing to Fog Computing," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 60–70, Apr. 2021. DOI: 10.48161/qaj.v1n2a49. [Online]. Available: https://journal.qubahan.com/index.php/qaj/article/view/49.

[11] Z. Meng, Z. Guan, Z. Wu, A. Li, and Z. Chen, "Security enhanced internet of vehicles with cloud-fog-dew computing," *Zte Communications*, vol. 15, no. S2, pp. 47–51, Dec. 2017. DOI: 10.3969/j.issn.1673-5188.2017.S2.008. [Online]. Available: http://kns.cnki.net/kcms/detail/34.1294.TN.20180102.1545.004.html.

[12] C. Inc., *Cisco locator/id separation protocol (lisp)*. [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/locator-id-separation-protocol-lisp/index.html.

[13] H. Chang, A. Hari, S. Mukherjee, and T. V. Lakshman, "Bringing the cloud to the edge," 2014, pp. 346–351. DOI: 10.1109/INFCOMW.2014.6849256. [Online]. Available: https://ieeexplore-ieee-org.vu-nl.idm.oclc.org/abstract/document/6849256.

[14] J. J. Bambara, P. R. Allen, K. Iyer, R. Madsen, S. Lederer, and M. Wuehler, "Introduction to blockchain," in *Blockchain: A Practical Guide to Developing Business, Law, and Technology Solutions*. New York: McGraw Hill Professional, Feb. 2018. [Online]. Available: https://pdfuni.com/sample/IT/IT1-100/IT055/sample%EF%BC%8DBlockchain%201st%201E%20Joseph%20Bambara.pdf.

[15] H. Vranken, "Sustainability of bitcoin and blockchains," *Current Opinion in Environmental Sustainability*, vol. 28, pp. 1–9, 2017, Sustainability governance, ISSN: 1877-3435. DOI: https://doi.org/10.1016/j.cosust.2017.04.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877343517300015.

[16] H. M. Kim, M. Laskowski, M. Zargham, H. Turesson, M. Barlin, and D. Kabanov, "Token economics in real life: Cryptocurrency and incentives design for insolar's blockchain network," *Computer*, vol. 54, no. 1, pp. 70–80, 2021. DOI: 10.1109/MC.2020.2996572. [Online]. Available: https://ieeexplore-ieee-org.vu-nl.idm.oclc.org/document/9321727.

[17] V. Puri, P. Kaur, and S. Sachdeva, "Data anonymization for privacy protection in fog-enhanced smart homes," in *2020 6th International Conference on Signal Processing and Communication (ICSC)*, IEEE, 2020, pp. 201–205.

[18] F. Y. Okay and S. Ozdemir, "A secure data aggregation protocol for fog computing based smart grids," in *2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018)*, IEEE, 2018, pp. 1–6.

[19] X. An, J. Su, X. Lü, and F. Lin, "Hypergraph clustering model-based association analysis of ddos attacks in fog computing intrusion detection system," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, pp. 1–9, 2018.

[20] S. Khan, S. Parkinson, and Y. Qin, "Fog computing security: A review of current applications and security solutions," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1–22, 2017.

[21] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Wireless Algorithms, Systems, and Applications: 10th International Conference, WASA 2015, Qufu, China, August 10-12, 2015, Proceedings 10*, Springer, 2015, pp. 685–695.

[22] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[23] M. Mukherjee, R. Matam, L. Shu, *et al.*, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017.

[24] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of things*, Elsevier, 2016, pp. 61–75.

[25] Y. Xiao and M. Krunz, "Dynamic network slicing for scalable fog computing systems with energy harvesting," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2640–2654, 2018.

[26] A. Chandak and N. K. Ray, "A review of load balancing in fog computing," in *2019 International Conference on Information Technology (ICIT)*, IEEE, 2019, pp. 460–465.

[27] G. Caiza, M. Saeteros, W. Oñate, and M. V. Garcia, "Fog computing at industrial level, architecture, latency, energy, and security: A review," *Heliyon*, vol. 6, no. 4, e03706, 2020.

[28] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Towards fault tolerant fog computing for iot-based smart city applications," in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*, IEEE, 2019, pp. 0752–0757.

[29] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–43, 2020.

[30] H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable fog computing," in *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, IEEE, 2013, pp. 43–46.

[31] J. Zarrin, H. Wen Phang, L. Babu Saheer, and B. Zarrin, "Blockchain for decentralization of internet: Prospects, trends, and challenges," *Cluster Computing*, vol. 24, no. 4, pp. 2841–2866, 2021.

[32] Y. I. Alzoubi, A. Al-Ahmad, and H. Kahtan, "Blockchain technology as a fog computing security and privacy solution: An overview," *Computer Communications*, vol. 182, pp. 129–152, 2022, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2021.11.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366421004321.

[33] Y. Liu, J. Zhang, and J. Zhan, "Privacy protection for fog computing and the internet of things data based on blockchain," *Cluster Computing*, vol. 24, no. 2, pp. 1331–1345, Oct. 2020. DOI: 10.1007/s10586-020-03190-3. [Online]. Available: https://link-springer-com.vu-nl.idm.oclc.org/article/10.1007/s10586-020-03190-3.

[34] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," *Future Generation Computer Systems*, vol. 101, pp. 747–759, Dec. 2019. DOI: 10.1016/j.future.2019.07.030. [Online]. Available: https://www-sciencedirect-com.vu-nl.idm.oclc.org/science/article/pii/S0167739X19301281?casa_token=YwN7JQoGS7QAAAAA:tzKlazZOj2ByWMKvPaIfGpT0Nd-6kVkj1rCZdot1csFksJV2GGqmkRZaNMpsLWxeT3AzAYapng.

[35] Y. Wu, G. Lu, N. Jin, L. Fu, and J. Zhuan Zhao, "Trusted fog computing for privacy smart contract blockchain," in *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*, IEEE, Oct. 2021. [Online]. Available: https://ieeexplore-ieee-org.vu-nl.idm.oclc.org/abstract/document/9688746?casa_token=sTwX1MS66ucAAAAA:bFsrDWPGPED8EWZQkXydCvQSLf_fh9rcGj75MaLGKZRf2oW8pvj0GUfR9qahx9ykGYH8sT0Utg.

[36] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019. DOI: 10.1109/TSMC.2019.2895123. [Online]. Available: https://ieeexplore-ieee-org.vu-nl.idm.oclc.org/abstract/document/8643084?casa_token=G4ShhFvOdHsAAAAA:mwuJVMkGWjVrcSmpWkHfYw3vB_6I3KYnMWenUBrrUMWhWuyrAlmro3aojsTozGspkCXKWd4wRg.

[37] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2901–2925, Apr. 2021. DOI: 10.1007/s12083-021-01127-0. [Online]. Available: https://link.springer.com/article/10.1007/s12083-021-01127-0.

[38] P. R. Nair and D. R. Dorai, "Evaluation of performance and security of proof of work and proof of stake using blockchain," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, IEEE, Feb. 2021. [Online]. Available: https://ieeexplore-ieee-org.vu-nl.idm.oclc.org/abstract/document/9388487?casa_token=SCMNOffxR78AAAAA:cIRLNmhO-o-3Z-pSzjCaX844964_MSu34bfV8a4M-OsZqMnkv_sjWnIgCh5oS8cbx3EuAsDYZ2Q.

[39] R. B. Uriarte and R. DeNicola, "Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 22–28, 2018. DOI: 10.1109/MCOMSTD.2018.1800020.

Cloud-Native, Distributed OLAP databases

Berry Chen
Department of Computer Science
University of Amsterdam
berry.chen@student.uva.nl

Weiqiang Guo
Department of Computer Science
University of Amsterdam
weiqiang.guo@student.uva.nl

Tianzheng Hu
Department of Computer Science
University of Amsterdam
tianzheng.hu@student.uva.nl

Abstract

This study aims to explore the characteristics and benefits of cloud-native OLAP databases, comparing them to on-premise OLAP databases. The paper begins by introducing OLAP and differentiating it from OLTP, emphasizing OLAP's capabilities in complex data analysis. The focus then shifts to the characteristics of cloud-native OLAP databases, which are optimized for cloud environments. Key features include scalability, elasticity, simplicity, performance, fault tolerance, and cost-efficiency. The paper further examines the advantages of using cloud-native OLAP databases over on-premise OLAP solutions. These benefits encompass enhanced simplicity, elasticity, and cost-efficiency through pay-as-you-go models and the cloud platform. In conclusion, the paper summarizes the key characteristics and benefits of cloud-native OLAP databases, emphasizing the advantages of leveraging cloud-native architectures for OLAP workloads.

1 Introduction

In recent years, there has been a remarkable shift in the way data is managed and stored, with a significant increase in the adoption of cloud databases. The rise of cloud computing has transformed the landscape of data storage and processing, offering unprecedented opportunities for organizations to harness the power of the cloud. The cloud's inherent features have made it a preferred choice for businesses across various industries. Within the realm of cloud databases, two prominent categories stand out: Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP). While OLTP databases excel in handling transactional workloads, OLAP databases are specifically designed for complex data analysis and decision-making.

The distinguishing factor that sets OLAP apart from OLTP is its ability to provide advanced analytical capabilities, enabling organizations to extract valuable insights from large volumes of data. OLAP databases are optimized for complex queries, aggregations, and multidimensional analysis, making them ideal for business intelligence, reporting, and data exploration. By leveraging OLAP, organizations can gain deeper insights into their data, identify trends, and make data-driven decisions that drive strategic growth.

In the context of cloud-native architectures, where applications and services are designed to fully harness the capabilities of cloud infrastructure, there is a growing interest in exploring the benefits of cloud-native, distributed OLAP databases. These databases are specifically tailored to operate seamlessly in cloud environments, taking advantage of the features offered by the cloud.

The main research question of this paper revolves around "**What are the key characteristics and benefits of cloud-native, distributed OLAP databases?**" By examining the specific features and comparing cloud-based OLAP to on-premise OLAP, we aim to shed light on how the OLAP databases provide valuable insights to the users and developers.

The rest of the literature study is presented as follows. Section 2 will give an overview of OLAP databases, comparing to OLTP databases. Section 3 will discuss the key characteristics of OLAP databases. Section 4 will present the implementation of on-premise OLAP databases and their drawbacks. Section 5 will introduce the implementations of cloud-native OLAP databases and the way they mitigate the drawbacks that om-premise databases have. Section 6 gives an experimental comparison between Vertica(an on-premise OLAP database) and Amazon Redshift(a cloud-native OLAP database). Section 7 will summarize the paper and answer our research questions. Lastly, Section 8 will address the future trends in OLAP databases field.

2 OLAP Introduction

2.1 Online Analytic Processing

Online analytic processing(OLAP) is a crucial part of decision support, which is gaining popularity in the industry. It enables managers or analysts to make better and faster decisions. The functional and performance requirements of OLAP databases are different from Online Transactional Processing(OLTP) databases. OLTP databases usually look up a small number of records by some key, using an index. It is typically designed for commercial transactions, such as making a sale, paying an employee's salary, etc. These tasks consist of atomic and isolated transactions. It tends to be hundreds of megabytes to gigabytes in size. In such a scenario, consistency and fault tolerance are crucial to the system. There are usually hundreds or thousands of users who access the databases concurrently. The databases should be deployed in the cluster so that it handles so many concurrent requests. It is a bad experience for users to get stale data or not be able to access the service normally. Hence, it is indispensable to handle consistency and faults when supporting users' activities. In addition, concurrency conflicts need to be minimized. Many users might send requests to the same place at the same time which might result in some conflict. In order to make it not harm the performance of the databases and data consistency, developers need to pay much attention to how to solve concurrency conflicts elegantly.

By contrast, an analytical query normally scans a large number of records, only reading a few attributes per record, and computes aggregate statistics metrics, such as count, sum, mean, median, or some more complex metrics, instead of returning a few records to the end users. It aims at helping managers or analysts to analyze the data and make better decisions. For example, if your data consists of sales transactions of a local supermarket, possible analytic queries might look like:

- Which kind of beer is most popular?
- How many potatoes we can sell every month?
- How many more apples than usual did we sells after we start the promotion?

The results of these queries are then analyzed by business analysts and put into reports to help the management department of a company to make better decisions. A normal company tends to have terabytes or petabytes of data to analyze. In OLAP applications, query throughput and response time are more important since the newer the results the decision maker gets, the more accurate the decision is. It normally read the data, hence it is rare to handle data consistency problems. The main difference between OLTP and OLAP is summarized in Table 1

Given that OLTP and OLAP databases have different access patterns and different performance requirements, performing analytical queries in OLTP databases would result in unacceptable performance and vice versa. Hence, most companies usually maintain two databases separately. However, there are many researchers trying to handle two different workloads in a single database, which they call HTAP databases[2].

Table 1: Comparing properties of OLTP versus OLAP systems[1]

Property	OLTP	OLAP
Read pattern	small number of records	aggregate over large number of records
Write pattern	Random access, low latency writes	batch import
Used by	End user/customer, via web application	internal analyst, for decision support
What data represents	Latest state of data	History of events that happened over time
Dataset size	Gigabytes or terabytes	Terabytes to petabytes

2.2 Data model

To mitigate the overhead of complex analytical queries, the data in the OLAP systems are modeled multi-dimensionally. In a multidimensional data model, there are a set of numeric measures called fact, such as sales, and budget. Each numeric measure depends on a set of dimensions, which provide the context for the measure. Taking sales as an example, the associated dimensions could be product name dimension, date dimension, etc. Each dimension is also described by a few attributes. For example, attributes for date dimensions could be the year, month, day, timestamp, etc. Most OLAP databases use star schema to represent the multidimensional data model. It consists of a single fact table and a single dimension table for every dimension. A few columns in a fact table are normally the foreign key to each of the dimension tables. An example of star schema is shown in Figure 1

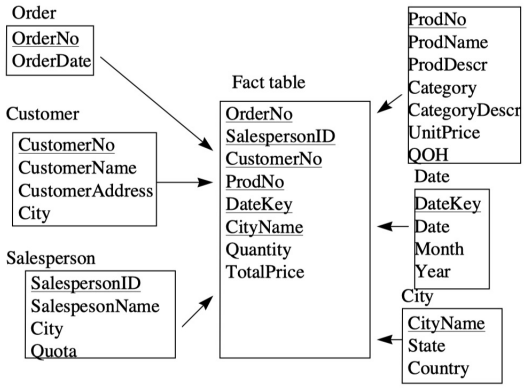


Figure 3. A Star Schema.

Figure 1: An example of star schema[3]

3 Properties

Scalability, elasticity, cost efficiency, simplicity and fault tolerance are some of the most important features of a typical cloud platform. OLAP databases offer high scalability and others key characteristics service, allowing organizations to handle increasing data volumes and user concurrency without compromising performance. In this section, these properties will be presented in detail.

3.1 Scalability and Elasticity

In the age of distributed computing, public cloud platforms can provide users with virtually unlimited computing and storage resources. Traditional databases have to adapt to the fact that users cannot afford the cost of traditional software-as-a-service and turn to take advantage of the scalability and elasticity of the cloud. Most of the mainstream high-performance databases are now shared-nothing architectures, due to their excellent scalability. This is because, in shared-nothing architectures, each

query processor node has its own local disk. Tables are split horizontally between the nodes and each node is only responsible for the rows on its local disk. This design scales well for star-structured queries because very little bandwidth is required to join a small (broadcast) dimension table to a large (partitioned) fact table[4].

Scalability refers to how much more load a system can handle for a given amount of allocated resources. A scalable system can handle a load that grows linearly according to the amount of resources available. Conversely, a poorly scalable system cannot handle more load even if it is given more resources. Scalability metrics typically include scalability range, resource scalability and cost scalability. Scalability range is the segment between the number of users that can be handled by a minimum deployment of resources and the maximum achievable capacity. Resource scalability is a function of the increase in capacity as a function of the increase in resources. Cost Scalability describes how capacity increases based on the minimum cost of the configuration, what the minimum deployed capacity is and the capacity of the deployment of increasing resources until the capacity plateaus.

There are some similarities between elasticity and scalability, with elasticity referring to the ability of a system to automatically increase or decrease its provision of resources to cope with the workload. It can usually be measured by counting the number of times the Service Level Objectives(SLO) rule is violated in a given period of time, or by finding the slope at which the load grows fastest. Examples include the average speed at which additional users can be processed without violating SLO criteria, and the average time it takes for a resilient system to return to normal SLO levels after a sudden increase in usage[5].

3.2 Performance

In cloud-native databases, performance is typically measured using various metrics that evaluate the efficiency and effectiveness of data processing and query execution. Improving a database's performance is essential to meet user expectations, enable faster decision-making, increase productivity, support scalability, optimize costs, gain a competitive advantage, and effectively power data-driven applications.

Some commonly used performance measurements for cloud-native databases, including query response time, query throughput, latency, resource utilization, and benchmark performance. These performance metrics provide insights into the efficiency, speed, and capacity of a cloud-native database. They help assess its ability to handle workloads, process queries, and deliver timely and accurate results. Monitoring and analyzing these metrics can guide performance optimization efforts and ensure that the database meets the desired performance requirements.

To enhance the performance of cloud-native OLAP databases, methods such as columnar storage, data compression, and vectorized processing are utilized.

Columnar storage is a key performance-enhancing feature of cloud-native OLAP databases. By organizing data in a columnar format rather than the traditional row-based format, these databases can achieve superior query performance. Columnar storage allows for efficient data compression and enables selective data retrieval, as only the required columns need to be accessed during query execution. This reduces I/O operations and improves overall query speed.

Data compression plays a crucial role in optimizing storage and query performance. Cloud-native OLAP databases employ advanced compression techniques tailored for columnar data. These techniques reduce the storage footprint and enhance I/O efficiency, resulting in faster data access and improved query response times. The use of compression also minimizes the network transfer time when moving data between nodes in distributed environments.

Vectorized processing is another performance-boosting technique employed by cloud-native OLAP databases. It enables operations to be performed on entire sets of data in a single instruction, known as SIMD (Single Instruction, Multiple Data) processing. By processing data in parallel, vectorized processing improves computational efficiency and accelerates query execution. This technique is particularly effective when working with large datasets and complex analytical operations.

3.3 Fault tolerance

Fault tolerance refers to a database's ability to continue functioning and providing access to data even in the presence of faults or failures. The goal of fault tolerance is to ensure the availability and reliability of the database, minimizing the impact of failures on data access and system operations. Handling fault tolerance is important because it ensures the continuous availability of critical systems, maintains data integrity and reliability, enables business continuity, enhances customer satisfaction, protects data and security, and reduces costs associated with failures and downtime.

The OLAP databases employ various mechanisms such as data replication, data durability techniques, automatic failure detection, and recovery mechanisms to achieve fault tolerance.

Data replication involves maintaining multiple copies of data across different nodes or instances within the cluster. This redundancy ensures that data remains available even if a node fails. By accessing replicas of the data, the system can continue to serve queries and process data seamlessly. [6]

Data durability is achieved through techniques such as write-ahead logging and persistent storage. Write-ahead logging ensures that data changes are first recorded in durable storage before being applied to the database. This allows for recovery in case of failures, ensuring data integrity and durability.

3.4 Simplicity

Simplicity is a key characteristic of cloud-native, distributed OLAP databases, and it refers to the ease of use, deployment, and management of these databases. Cloud-native OLAP databases prioritize simplicity by leveraging various architectural paradigms such as Software-as-a-Service (SaaS) and serverless computing. These approaches contribute to the ease of use and management of the databases.

Cloud-native OLAP databases offer intuitive and user-friendly interfaces that simplify the process of querying and analyzing data. They often provide visual query builders, drag-and-drop functionalities, and interactive dashboards, enabling users to interact with data in a self-service manner without requiring extensive technical expertise.

SaaS models further enhance the simplicity of cloud-native OLAP databases. With SaaS, users can access the database through a web browser, eliminating the need for complex installations or configurations. This not only reduces the upfront setup effort but also ensures that users are always using the latest version of the database without needing to perform manual updates. SaaS also enables easy collaboration and sharing of analytical insights by providing access to the database from anywhere with an internet connection.[7]

Serverless computing takes simplicity to the next level by abstracting away infrastructure management. In a serverless environment, users do not have to worry about provisioning or scaling resources. The database provider takes care of automatically scaling the infrastructure based on the workload, ensuring optimal performance and eliminating the need for manual intervention. This allows users to focus solely on their analytical tasks without being burdened by the complexities of infrastructure management. For example, with the automatic on-demand provisioning and scaling capabilities of Amazon EMR Serverless, Kyligence can quickly meet changing processing requirements at any data volume.[8]

3.5 Cost-efficiency

The cost-effectiveness of a cloud database is based on its inherent elasticity and scalability characteristics. Scalability allows users to be as cost-efficient as they need to be and is an important aspect to consider when evaluating a cloud database solution, referring to the balance between minimum cost and maximum performance of the database service and how to maximize the efficiency of the investment. Elasticity further improves cost efficiency by automatically increasing or decreasing resources based on real-time demand, allowing businesses to pay only for the resources they need at any given time.

Cloud database providers need to consider the cost of transferring and storing data, and optimising query performance and database design can often reduce costs. An effectively designed database,

with proper indexing, partitioning, and data organization, can improve the query performance of data, thereby reducing resource usage and ultimately cost.

4 Existing on-premise OLAP databases

On-premise OLAP databases are deployed in the on-premise cluster to handle analytical queries, which help decision-makers make better and faster decisions. In this section, we cover several popular OLAP databases which are commonly used in industry and describe each.

Vertica[9] The Vertica Analytic database(Vertica) is a commercial, columnar-oriented, distributed, Massively Parallel Processing(MPP) database supporting efficient analytical workloads on large datasets while still providing classic relational interfaces. It was founded in 2005 by Michael Stonebraker with Andrew Palmer and then widely used in business intelligence systems. It originated from the C-Store column-oriented DBMS[10], an open-source project created by MIT and other universities in 2005.

Greenplum Greenplum is a distributed MPP database focusing on analytical workloads based on PostgreSQL[11] technology. It was created in around 2005 in California, USA. It recently aims to be a hybrid system that supports both OLAP and OLTP workloads[12].

Druid[13] Druid is a column-oriented, open-source, distributed OLAP database that is designed for analyzing a large amount of real-time and historical data. It was created in 2011 to help the analytics product of Metamarkets. It leverages columnar storage, shared-nothing architecture, and advanced index techniques to speed up analytical queries. It provides several powerful features, such as low latency data ingestion, arbitrary slice and dice data exploration, etc.

Krylin Krylin is an open-source distributed OLAP engine started by eBay in 2013. It is designed on top of Apache Hadoop, Apache Hive, Apache HBase, and Apache Parquet to support the functionality of analyzing extremely large data in low latency. It was widely adopted by many industry technology companies, such as eBay, Yahoo!Japan, etc.

Almost all of these implementations have a few things in common. First, they leverage columnar storage which stores data column by column, instead of storing data row by row. This idea is inspired by the access pattern of OLAP workloads. Analytical queries tend only to access a few columns of a table. Hence, there is no need to read all columns into memory. In addition, data in the same column tend to have the same type, even fixed range which makes it possible to compress the data. These techniques can reduce the I/O overhead significantly and as a result, speed up the query throughput and performance remarkably. Second, some advanced index techniques optimized for analytical workloads are deployed to enhance the query performance further. Ultimately, they are all distributed systems leveraging multiple servers to perform tasks that are deployed in on-premise clusters. Analytical workloads need to scan an extremely large number of data which is too expensive, and even not possible to be handled by a single machine.

However, the on-premise OLAP databases have several serious drawbacks,

1. It is NOT Simple. If a user would like to exploit OLAP databases to extract insights from the data, he needs to buy a cluster, install OLAP databases, and then configure and operate them. In addition, security and authentication also require attention. It would take a lot of effort.

2. It is NOT Elastic. If a user would like to resize the databases, for example, by adding a node, he might need to buy a new machine and configure it. It is obviously this would take time. In addition, removing a node might result in wasting resources.

3. It is NOT cost-efficient. Users normally need to configure the databases according to the requirements of the peak time in order to guarantee the service is always available. However, peak time rarely happens in reality, which means that users are wasting their resources most of the time.

The needs for solving these drawbacks drive people to build cloud-native OLAP databases which leverage the power of cloud computing.

5 Existing Cloud-native OLAP databases

In this section, we present the Cloud-native OLAP databases and their implementations. In addition, how cloud-native OLAP databases can mitigate the drawbacks of on-premise OLAP databases is also presented.

Amazon Redshift[14] Amazon Redshift is the first mover in cloud data analysis systems, which is based on the columnar analytical database Paraccel[15]. It is a fast, fully-managed OLAP database that aims to provide a simple and cost-effective way to analyze a large volume of data. It leverages techniques that are commonly used in the OLAP field, including columnar layout, data compression, efficient operators, and code generation. Using these mature techniques and a series of optimization methods, the performance of Redshift is comparable to the traditional columnar analytical databases.

In terms of simplicity, Amazon Redshift aims to achieve Software-as-a-service(SaaS) that mitigates the complexity of users' experience. First, Redshift offers database administration functionalities, such as provisioning, backup, monitoring, etc. Users could focus on the analysis of the data and their business logic instead of spending much time and money on managing their databases. Apart from that, Amazon Redshift also provides some tuning knobs which removes the burden of tuning databases from users. The main things users need to consider are setting the type and number of nodes for the cluster and the sort and distribution model. In this way, the complexity of managing databases is significantly reduced and as a result, it is simple for users to analyze their data.

In terms of cost efficiency, unlike traditional on-premise OLAP databases, which require large upfront payments, Amazon Redshift offers the pay-as-you-go cost model. Users pay only for compute and storage capacity consumed while processing their workloads. When they finish their workloads, they can simply shut down the cluster and save their money, which is hard to achieve in on-premise clusters. In the on-premise cluster, the company needs to configure the resources according to the requirements of peak time which rarely appears in order to meet the needs of end users. They have to waste resources and money during low peaks. Cloud-native databases overcome this disadvantage and make it cost-efficient by leveraging the power of cloud computing.

In terms of elasticity, Amazon Redshift leverages the Amazon Elastic Compute Cloud(EC2) and Amazon S3. It can scale out the cluster automatically to meet the performance requirement. It is deployed in shard-nothing architecture that every node has private CPUs and disks and communication with each other via the network.

In terms of fault tolerance, Amazon Redshift relies on Amazon S3 which is highly available, and durable. It also replicates the data to enhance the availability and fault tolerance. If one piece of data is lost, users can access another replica.

Snowflake[4] Snowflake is a more recent cloud-native data warehouse system, developed especially for cloud platforms. It shares many design ideas with Amazon Redshift, including column-oriented storage, common techniques used in OLAP databases, SaaS service, and the pay-as-you-go cost model which make it performant, elastic, fault-tolerant, and cost-efficient. However, it is more elastic and scalable than Amazon Redshift. Unlike Amazon Redshift is deployed in the shard-nothing architecture, Snowflake is deployed in the shared-data architecture. More specifically, it provides computing via shared-nothing architecture where local disks act as a cache and storage through Amazon S3. Shared-nothing architecture is the dominant choice in high-performance OLAP databases since it is highly scalable. Nevertheless, this architecture has a serious shortcoming, that is, the compute resources are tightly coupled with storage resources, which results in some problems. For example, if the number of nodes in a cluster changes, either because users want to resize the cluster, or due to node failures, the data needs to be redistributed which would consume a large amount of time. In the shared-data architecture, the compute and storage are able to scale independently which makes the system more elastic.

Apart from elasticity, Snowflake also adds more support for semi-structured and unstructured data. As the need for machine learning increases, leveraging semi-structured and unstructured data becomes more and more important.

An interesting finding from Snowflake is that performance turns out not to be a problem for most users due to the high elasticity of the system. Users could add more nodes to boost performance instead of optimizing the internal implementation, which makes them could focus on other important issues.

BigQuery Google BigQuery originated from the Google Dremel[16] is designed for analytical workloads. It provides a higher level of abstraction than Amazon Redshift and Snowflake. Unlike Amazon Redshift and Snowflake, it is serverless[17]. Amazon Redshift and Snowflake aim to offer Software-as-a-service service, however, users still need to set up and configure the databases clusters. For example, users need to choose the type and number of nodes in their clusters. By contrast, there are no such notions in Google BigQuery. What users need to do is upload/choose the data and write the query statements, and then BigQuery will run the query itself. Users do not need to know the underlying systems and then could focus more on their business logic.

In summary, cloud-native OLAP databases generally use similar methods to achieve performance and fault tolerance. More importantly, they deliver Software-as-a-service and serverless service which removes the burden of configuring and operating databases from users in order that they can focus on their business logic and make the systems simpler to use, more elastic and more cost-efficient.

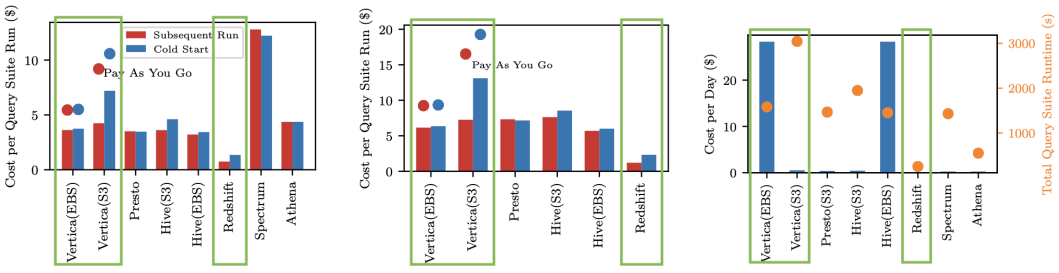
6 Comparison

In order to provide the consumption cost comparison between on-premise and the cloud database, Vertica and Redshift have been selected as two examples.

Elastic Block Store (EBS) and the Simple Storage Service (S3) are two kinds of storage services provided by AWS. EBS is a remote network storage option that uses a standard file system API. It offers both SSD (Solid State Disk) and HDD (Hard Disk Drive) options. It also provides persistent storage that remains even when the compute node is shut down, which makes EBS a suitable choice for DBMSs (Database Management Systems). On the contrary, S3 is an object store that operates on specialized S3 nodes and is accessible via a web-based REST API. It is intended for scalable and high-concurrency applications, however, it has higher latency and more unpredictability in throughput than block storage. Both EBS and S3 are used in Vertica, while Redshift is AWS's parallel processing OLAP database service that leverages a typical shared-nothing architecture. It is accessible in instance sizes that are not labeled the same as the compute types available on EC2, therefore it may be coupled with additional hardware that is not available to common users.

The system evaluation is characterized between database-as-a-service offerings and cloud provider agnostic OLAP DBMS, which are Redshift and Vertica individually. The system settings start as "cold cache" to "warm cache". In the "cold start" case, researchers followed the all vendor instructions for clearing the DBMS and OS caches after each query execution to avoid caches effect. For Redshift, they also even restarted the system or recreated the clusters respectively. In the "warm cache" case, researchers firstly ran the whole query after the cold start before taking further measurements on subsequent executions.

No matter whether Vertica uses EBS or S3, Redshift always costs much less in computing than Vertica as shown in Figure 2a and Figure 2b. If Vertica uses EBS as a storage database, it will cost a lot, but if it uses S3, the investment will be much lower as shown in Figure 2c. However, no matter which storage Redshift uses, it spends the least amount of money.



(a) Compute cost on all systems (b) Compute cost on subset of systems (c) Storage cost

Figure 2: System cost comparsion[18]

7 Conclusions

The purpose of this research is to investigate the features and benefits of cloud-native OLAP databases in comparison to on-premise OLAP databases. Key properties such as scalability, elasticity, simplicity, performance, fault tolerance, and cost-efficiency are studied. From a developer's point of view, we believe that simplicity is the most important feature to consider, followed by scalability and performance. The properties bring cloud-native OLAP databases a lot of benefits over traditional OLAP. By comparing the on-premise databases and cloud-based databases, cloud-based OLAP databases are particularly advantageous in terms of the additional cost.

8 Future

OLAP databases are used by more and more companies to analyze the data produced by their daily activities and help them make faster and better decisions. It turns out that it is not replaced by Big data technologies such as Hadoop and Spark, but it has complimented them. As the need for analytical workloads increases continuously, the OLAP databases also keep evolving in order to meet the requirements of new trends. By reviewing all papers mentioned in this article, we have identified three main trends that seem to be crucial and would be likely to get more attention in the future: HTAP databases, more support for semi-structured and schema-less data, and being a full self-service model.

The first branch of the future work we identified is supporting both transactional and analytical workloads in a single database. Nowadays, companies operate two separate systems for handling transactional and analytical workloads. The data in the OLTP databases is transferred to OALP databases and then analyzed by the data scientists, which obviously incurs many management overheads. As a result, there is growing interest in handling both workloads in a single system although it is a hard problem. The difficulty of merging two kinds of databases is that they have different access patterns and requirements. OLTP databases focus on low latency and concurrency conflicts. However, OLAP databases focus more on high throughput. There are already some solutions for this problem, such as TiDB [19] which is based on the Raft consensus algorithm.

The second branch of improvement is adding more support for semi-structured and schema-less data. In the age of machine learning, semi-structured and schema-less data are playing a more and more important role in data analysis. There is a fast-growing need in processing these data and extract useful information.

The third branch of future research is that make the OLAP databases a self-service model. Unlike in traditional OLAP databases, users need to configure the clusters and databases themselves, cloud-native OLAP databases allow users to focus on their business logic instead of configuring the databases. However, in some circumstances, users still need to interact with underlying systems, such as configuring the parameters of databases to improve performance, asking for the help of service providers as a result of encountering some security issues, etc. More and more cloud providers are searching for methods to automatically configure systems and achieve high performance concurrently by leveraging machine learning techniques, such as Qtune[20], and also putting more attention on security, and performance problems.

References

[1] Martin Kleppmann. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. " O'Reilly Media, Inc.", 2017.

[2] Fatma Özcan, Yuanyuan Tian, and Pinar Tözün. Hybrid transactional/analytical processing: A survey. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1771–1775, 2017.

[3] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[4] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, et al. The snowflake elastic data warehouse. In *Proceedings of the 2016 International Conference on Management of Data*, pages 215–226, 2016.

[5] Sebastian Lehrig, Richard Sanders, Gunnar Brataas, Mariano Cecowski, Simon Ivanšek, and Jure Polutnik. Cloudstore—towards scalability, elasticity, and efficiency benchmarking and analysis in cloud computing. *Future Generation Computer Systems*, 78:115–126, 2018.

[6] Hemant Saxena and Jeffrey Pound. A cloud-native architecture for replicated data services. In *Proceedings of the 12th USENIX Conference on Hot Topics in Cloud Computing*, pages 19–19, 2020.

[7] Catalin Strîmbei. Olap services on cloud architecture. *Journal of Software and Systems Development*, 2012:1, 2012.

[8] Daniel Gu, Yolanda Wang, and Kiran Guduguntla. How kyligence cloud uses amazon emr serverless to simplify olap, Nov 2022.

[9] Andrew Lamb, Matt Fuller, Ramakrishna Varadarajan, Nga Tran, Ben Vandier, Lyric Doshi, and Chuck Bear. The vertica analytic database: C-store 7 years later. *arXiv preprint arXiv:1208.4173*, 2012.

[10] Mike Stonebraker, Daniel J Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, et al. C-store: a column-oriented dbms. In *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker*, pages 491–518. 2018.

[11] Bruce Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley New York, 2001.

[12] Zhenghua Lyu, Huan Hubert Zhang, Gang Xiong, Gang Guo, Haozhou Wang, Jinbao Chen, Asim Praveen, Yu Yang, Xiaoming Gao, Alexandra Wang, et al. Greenplum: a hybrid database for transactional and analytical workloads. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2530–2542, 2021.

[13] Fangjin Yang, Eric Tschetter, Xavier Léauté, Nelson Ray, Gian Merlino, and Deep Ganguli. Druid: A real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 157–168, 2014.

[14] Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, and Vidhya Srinivasan. Amazon redshift and the case for simpler data warehouses. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1917–1923, 2015.

[15] Yijou Chen, Richard L Cole, William J McKenna, Sergei Perfilov, Aman Sinha, and Eugene Szedenits Jr. Partial join order optimization in the paraccel analytic database. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 905–908, 2009.

[16] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2):330–339, 2010.

[17] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*, 2019.

[18] Junjay Tan, Thanaa Ghanem, Matthew Perron, Xiangyao Yu, Michael Stonebraker, David DeWitt, Marco Serafini, Ashraf Aboulnaga, and Tim Kraska. Choosing a cloud dbms: architectures and tradeoffs. *Proceedings of the VLDB Endowment*, 12(12):2170–2182, 2019.

[19] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, et al. Tidb: a raft-based htap database. *Proceedings of the VLDB Endowment*, 13(12):3072–3084, 2020.

[20] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. Qtune: A query-aware database tuning system with deep reinforcement learning. *Proceedings of the VLDB Endowment*, 12(12):2118–2130, 2019.

Big data and Cloud in Medical Industry
Group 8

Yuna Chen
Department of Computer Science
University of Amsterdam
14274310
yuna.chen@student.vu.nl

Shutong Cai
Department of Computer Science
University of Amsterdam
14730960
shutong.cai@student.uva.nl

Yudong Fan
Department of Computer Science
University of Amsterdam
14729024
yudong.fan@student.uva.nl

Abstract

With the rapid development of technologies, digital data are generated and collected at an unprecedented rate and scale. In recent times, the popularity of "Big Data" and "Cloud" concepts and their applications has been steadily increasing, causing significant transformations in numerous industries away from their conventional norms. Particularly, medical industry, which is one of the domains that has been actively adopting and embracing a digital evolution based on the rise of big data and cloud technologies. Many novel applications and opportunities emerge alongside the evolution, as well as a series of challenges. This paper intends to answer one research question: How do big data and cloud impact modern medical industry? To answer this question, this paper provides a high-level overview of the big data and cloud technologies including the challenges they impose within the medical industry. Furthermore, big data and cloud in two specific contexts in medical industry, healthcare and biomedicine, are discussed thoroughly to provide a more intuitive entry point to comprehend and answer the research question.

Keywords: Big data, Cloud, Medical Industry, Healthcare, Biomedicine, Parallel Computing, Security, Privacy

1 Introduction

Alongside the flood of digital data from a variety of digital earth sources such as sensors, smart devices, Internet, etc., the idea of big data has become more and more important as it provides angles to improve and/or enable research and decision-making with unprecedented value for various digital earth applications [40]. That being said, the term "big data" is a rather vague and broad concept. By studying existing definitions of big data, Mauro et al. [12] proposes a nucleus definition for big data. That is, "Big Data represents the information assets characterized by such a high volume, velocity and variety to require specific technology and analytical methods for its transformation into value." Moreover, according to Hashem et al., [20], big data can be characterized by three aspects: 1) data are numerous; 2) data cannot be categorized into regular relational databases; 3) data are constantly and rapidly generated, captured, and processed. Given these characteristics, big data presents challenges to researchers from many perspectives including data storage, transportation, process, etc. As a matter

of fact, the emerging trend of big data is transforming many industries and even society from their traditional profile. Particularly, the medical industry is one of the disciplines which is increasingly adopting big data technologies. Nowadays, enormous amounts of biological and clinical data are generated and collected at an unprecedented speed and scale [26]. As a result, a series of relevant challenges and opportunities also emerge within the field.

Cloud computing plays a crucial role in solving challenges that big data imposes. It makes possible for industrial applications to perform large-scale and complex computing. Not only because it enables virtualization, parallel processing, security, scalable data storage and service, but also because it reduces the cost and restriction for automation and computerization by individuals and enterprises [25]. However, yet another major challenge is that the growth rate of big data exceeds the current technology capability both in terms of properly designing cloud computing platforms and updating intensive workloads [20]. In the medical industry domain, the challenges of storing, managing, and analyzing massive medical data have been drawing researchers' attention for many years [26]. Due to the characteristics of big data as discussed above, powerful big data and cloud computing technologies are indispensable to facilitate the revolution for various applications within the industry.

This study intends to focus on two specific segments in the medical industry domain. Namely, the biomedicine and healthcare systems. By carrying out a comprehensive investigation regarding the current big data and cloud applications, challenges, opportunities, and concerns, we aim to provide a heuristic and intuitive entry point to answer the research question about how big data and cloud impact the medical industry.

The rest of this paper is conducted as follows. Section 2 gives an overview of the commonly used big data and cloud technologies in the modern medical industry, as well as the challenges these technologies impose. Section 3 and Section 4 elaborate on how big data and cloud technologies are used specifically regarding biomedicine and healthcare segments respectively within the general medical industry. More detailed contextual investigations are engaged, and possible solutions to cope with the potential technological and ethical challenges are proposed. Section 5 initiates a comprehensive discussion regarding the common pitfall and dilemmas, and Section 6 concludes the entire paper.

2 Big data technologies and challenges in Medical Industry

The medical industry as a whole is a multi-dimensional system which aims to prevent, diagnose, and treats health-related issues or impairments in human beings. Due to the sensitive nature and complex composition of this industry, the big data repository for medical data is highly diversified and requires advanced techniques to ensure relatively high-level security, correctness, relevancy, consistency, stability, and accuracy regarding data storage, privacy, sharing, and processing [11]. Given the high volume, velocity, and variety of characteristics of big data, the industry has adopted different big data technologies to fulfil its essential requirements, which subsequently gave rise to a series of challenges regarding some specific concerns in the medical industry.

2.1 Parallel Computing and Distributed Data Storage

Parallel computing is one of the key components to handle big data. The essence of parallel computing is to distribute and process data across multiple nodes. This relates to both data storage and processing frameworks as traditional monolithic data storage and processing technologies are no longer sufficient or even feasible to handle big data in an effective and secure way. One of the most popular open-source distributed applications is Hadoop [37]. In short, Hadoop is an implementation of the MapReduce algorithm, which in essence tries to split a big problem into smaller ones by performing the mapping procedure to create intermediate key/value pairs, and then solve the smaller problem in parallel and merge the result based on the same keys via the reducing procedure [14]. It runs on top of the Hadoop Distributed File System (HDFS) which is a distributed file system that provides a scalable and fault-tolerant infrastructure to store data in a distributed environment. Moreover, because it enables redundant replicas of data chunks, it is less prone to data loss. Consequently, technologies which enable parallel computing like Hadoop and its alternatives (e.g., Apache Spark) make it possible to efficiently store and process big data in a distributed environment (i.e., a large-scale cluster consisting of many machines). Nevertheless, such technologies also have its drawback. For instance, despite the community has been actively upgrading Hadoop's security mechanism, it still exposes security

vulnerabilities which might be exploited by malicious activities and potentially cause serious harm [32]. This remains to be a key challenge for researchers in the medical industry because security is one of the utmost important requirements for medical data.

2.2 AI-based Information Processing

Nowadays, machine learning and AI techniques becomes more and more popular to refine the information processing capabilities in the medical industry. For example, the rise of natural language processing (NLP) technology eases the process of generating, querying, retrieving, and analyzing textual documents such as clinical notes [11]. Moreover, predictive machine learning models can even learn from massive input and help physicians to make better clinical decisions or even replace human judgement in certain functions. Nevertheless, sophisticated AI model tends to entail poor interpretability because the underlining operations are usually black-box operations. Therefore, concerns from both technical and ethical aspects for the credibility and security of AI-based data processing techniques keep drawing people's attention.

2.3 Concerns for Cloud

Notably, the big data technologies as discussed above are commonly associated with cloud environments under the current context because cloud providers provide on-demand usage of resources and therefore minimize the cost and restrictions for big data applications [25]. However, this major advantage of the cloud shifts into a disadvantage when it comes to the medical industry. From a medical data perspective, many organizations are more comfortable with data storage on their own premises because it guarantees control over security, access, and up-time. Since medical data are very sensitive, even though an on-site cluster is more expensive to scale and maintain, medical organizations still tend to show low willingness in terms of giving away their complete control over their data to a third-party could provider [11]. Therefore, how to bridge this trust gap between medical organizations and cloud providers and facilitate a cloud-based ecosystem for medical data become a primary challenge that is worth studying.

3 Big Data and Cloud in Biomedicine

The field of biomedicine has emerged as a groundbreaking interdisciplinary subject that integrates the principles and techniques of medicine, life science, and biology. Its primary objective is to apply biological and engineering methods to address and explore issues related to life science, particularly in the realm of medicine. Biomedical information, an integral component of the vast domain of medical "big data," plays a pivotal role in shaping and advancing biotechnology in the 21st century. It serves as a crucial engineering field that is instrumental in elevating the standards of medical diagnosis and overall human health [31].

This section begins by discussing the implications and difficulties arising from the use of large-scale data in the field of biomedicine. Subsequently, we delve into the introduction of cloud-based solutions, focusing on the various services and systems available within the cloud-based paradigm.

3.1 Big Data applications in Biomedicine

The global big data market in healthcare is projected to exceed $70 billion USD by 2025, with the United States expected to dominate over 90% of the North American market [8]. The abundance of healthcare data, including patient-specific clinical data is one of the main drives of this exponential growth. Organizations are using analytical tools, artificial intelligence (AI), and machine learning (ML) techniques to extract data-driven insights, leading to reduced costs, enhanced revenue streams, personalized medicine, and proactive patient care.

Big data integration is transforming biomedical research by consolidating data from laboratory experiments, clinical investigations, healthcare records, and the Internet of Things. Omics technologies, such as genomics, epigenomics, transcriptomics, proteomics, metabolomics, and pharmacogenomics, are generating vast amounts of information, presenting opportunities for advancements in personalized medicine and improved patient care [7]. The ultimate objective is to establish personalized medicine programs that significantly enhance patient care. By applying big data and advancing our

understanding of omics information, researchers aim to identify causal genetic factors, enabling tailored treatment approaches based on the right target, chemistry, and patient. This comprehensive approach holds great potential for revolutionizing patient outcomes. The field of biomedicine is experiencing significant advancements through the integration of omics technologies into big data analysis. This progress has profound implications for disease diagnosis and patient care management. Omics big data, particularly in research and healthcare, is primarily focused on enhancing diagnostic capabilities [10] [19].

However, the most remarkable contribution of big data lies in its pivotal role in advancing AI and machine learning techniques. These advancements have significant applications in the development of biomedicine, which holds immense potential for transforming the future of healthcare delivery. The convergence of big data and biomedicine has opened up unprecedented opportunities in the field, particularly in personalized medicine, where tailored treatments can be designed based on individual characteristics and needs. This convergence is poised to reshape the landscape of healthcare, ultimately leading to improved patient outcomes and enhanced healthcare practices.

Misdiagnosis is a significant challenge in biomedicine and healthcare, as it hinders patients from receiving appropriate treatment and care [8]. The current diagnostic methods, reliant on manual reviews by hospital staff and physicians, are error-prone. Diagnostic errors are complex, elusive, and difficult to measure, with no clear threshold for identifying them. To address this issue, researchers and clinicians at the Johns Hopkins School of Medicine have introduced a novel approach known as Symptom-Disease Pair Analysis of Diagnostic Error (SPADE) [24]. This method harnesses big data and AI algorithms to mine extensive clinical databases, encompassing numerous patient visits. By identifying common clinical symptoms associated with doctor visits, SPADE enables the pairing of these symptoms with diseases that are prone to misdiagnosis in specific clinical contexts. SPADE is particularly effective for acute and subacute diseases, which frequently result in hospitalization or disability.

The field of biomedicine and healthcare is undergoing a transformation with the integration of AI and ML techniques. While traditional statistical methods have limitations in handling complex biological datasets, ML algorithms offer powerful tools for analyzing big data and identifying meaningful patterns and features [8]. ML techniques such as principal component analysis, graph-based clustering, random forests, and deep learning have been successfully applied to genomics data, aiding in the understanding of molecular details and disease states [26]. This approach holds great promise for improving disease diagnosis, treatment, and prevention through customized approaches tailored to individual patients.

3.2 Challenges in Biomedical Big Data

The advent of big data generation and acquisition has sparked a paradigm shift, posing significant challenges pertaining to the storage, transfer, and security of information [7]. The cost associated with generating data has now become comparatively lower than that of storing, securing, and analyzing it. Notably, biological and medical data exhibit greater heterogeneity than data from other research domains. Analysis reveals that biomedical data is characterized by its voluminous nature and the inherent difficulties of handling it [41].

The rapid growth of clinical data and advancements in AI technologies have presented researchers with unprecedented challenges in managing data storage, scalable processing, and analysis capabilities for diverse and multi-sourced datasets. One of the primary obstacles lies in the necessity for substantial computational power and storage capacity [13]. The processing and analysis of big data necessitate extensive resources that often surpass the capabilities of individual institutions or researchers. Moreover, the quality and accuracy of data used in machine learning models are pivotal in ensuring reliable predictions. However, biomedical data can be afflicted with noise, incompleteness, or bias, thereby potentially influencing the performance of machine learning algorithms.

Another challenge have to mention lies in the efficient transfer of data between locations. All transfer protocols encounter obstacles when dealing with the transfer of large, unstructured datasets. Consequently, tools that expedite the data transfer process and mitigate latency issues are imperative.

The security and privacy of individual data constitute an additional concern. Biomedical research heavily relies on access to vast amounts of sensitive information. Patient data, stored in electronic form within medical databases and bio-repositories, need to be queried, mined, and analyzed by

doctors and researchers [9]. In light of the collaborative nature and the imperative for data sharing, it becomes crucial to address data security and privacy concerns.

3.3 Cloud Adoption in Biomedicine

The challenges associated with ensuring secure data storage, establishing effective data integration models, and implementing efficient data analysis methods can be effectively tackled through the utilization of high-performance computing (HPC) and cloud solutions. Parallel computing and cloud computing present orthogonal yet highly efficient and scalable solutions to address these challenges. In practical terms, the utilization of parallel biomedical software on widely available high-performance computers can be applied at a lower layer to preprocess and analyze biomedical data [3]. While numerous data analysis tools have been developed in the field of biomedicine, only a few have been adapted to harness the full potential of high-performance computing resources [23]. For certain tools, an appealing option is to employ a map/reduce strategy, allowing for improved scalability and performance.

In contrast, cloud computing offers extensive capabilities for data storage, data sharing services, and the flexibility of accessing resources and applications on-demand, regardless of time and location. Furthermore, HPC tools can be executed in parallel on cloud platforms, further enhancing their scalability and adaptability. This enables the development of elastic and scalable applications and services in the context of biomedicine. It is noteworthy that the integration of HPC tools with cloud platforms provides researchers and practitioners with enhanced computational power and efficient data handling, thereby facilitating large-scale applications in biomedicine. In this section, our focus is specifically on cloud-based biomedicine services and systems that are specifically designed to address the demands of large-scale applications.

3.3.1 Cloud-based Services in Biomedicine

Over the past decade, cloud computing has gained significant traction in the healthcare and biomedicine domains, finding extensive adoption in both academic and industrial settings [5]. The Cloud computing model has witnessed a rapid proliferation in recent years as a means of delivering IT resources, encompassing hardware and software, through network-accessible services. This model offers several novel advantages, including access to massive and scalable computing resources on-demand, the utilization of virtualization technology, and a pay-as-you-go pricing model based on resource usage [3]. In the realm of biomedicine, Cloud-based services can be categorized into four main models, which are outlined in the subsequent content.

Data as a Service (DaaS): The conventional approach to biomedical pipeline analysis involves downloading or accessing public datasets from repositories such as NCBI or Ensembl, installing software locally, and performing in-house analyses. However, transitioning to a cloud-based environment offers enhanced integration capabilities that optimize the analysis and storage of biomedical big data. DaaS provides the dynamic virtual space provided by the cloud to facilitate data storage, ensuring that the data remains up-to-date and accessible from a diverse range of connected devices via the web. Notably, Amazon Web Services (AWS) offers a notable example of DaaS with its centralized repository of public datasets. This repository includes archives of prominent databases such as GenBank, Ensembl, the 1000 Genomes Project, Unigene, and Influenza Virus. AWS provides these datasets as public services within their cloud infrastructure, allowing for seamless integration into cloud-based applications.

Software as a Service (SaaS): In recent years, there has been a proliferation of cloud-based tools, known as Software as a Service (SaaS), developed for various biomedical tasks, including genomics analysis, sequence alignment, and gene expression analysis. These tools often use the parallel execution capabilities of the open-source Hadoop implementation of MapReduce, which allows for distributed processing across multiple compute nodes. One example is ProteoCloud [29], a comprehensive cloud-based platform in the field of proteomics, offering five different peptide identification algorithms and an easy-to-use graphical user interface. Another example is CloudBurst which stands out as a parallel read-mapping algorithm optimized for mapping next-generation sequencing (NGS) data to reference genomes in the genomic domain [34]. These examples highlight the increasing

utilization of cloud-based platforms in biomedicine, which provide scalable and parallel processing capabilities, improving the efficiency and performance of various analyses in biomedical big data.

Platform as a Service (PaaS): In the field of biomedicine, a Platform as a Service (PaaS) model provides users with the flexibility to customize their solution deployments and exert full control over their instances and associated data, distinguishing it from SaaS. Currently, several Cloud-based platforms cater to the needs of researchers in this domain. Examples include CloudMan [2], which empowers individual biomedical researchers to effortlessly deploy, personalize, and share their complete datasets, tools, and configurations. Another prominent biomedical PaaS offering is Eoulsan [22], which is based on the Hadoop implementation of the MapReduce algorithm for high-throughput sequencing data analysis. It provides a dedicated environment for processing large-scale sequencing datasets efficiently. PaaS solutions exemplify the range of Cloud-based platforms available to researchers, enabling them to tailor their computational workflows, streamline data analysis, and facilitate collaboration and knowledge sharing in the field of biomedicine.

Infrastructure as a Service (IaaS): Within the field of biomedicine, the IaaS model contains a computing infrastructure that encompasses servers, often virtualized, with dedicated computational capabilities and storage resources. An example illustration of bioinformatics IaaS is BioNimbus [21], an open-source cloud-computing platform specifically designed to process genomics and phenotypic data. BioNimbus has been tailored to accommodate next-generation sequencing instruments and incorporates cutting-edge technologies for efficient analysis and the seamless transfer of large datasets. IaaS solutions exemplify the potential of cloud computing in facilitating the computational infrastructure required for biomedical data analysis. By employing virtualized servers and flexible storage resources, these platforms offer researchers the means to efficiently process and analyze large-scale datasets, ultimately driving advancements in the field of biomedicine.

3.3.2 Cloud-based Ecosystems for Biomedical Research

As was shown in the part before, using cloud-based services with different models offers a workable solution to the problems related to data storage and analysis. We will elaborate on the cloud-based ecosystem solution in this section, which gives institutions and researchers the opportunity to improve the chances of open science in the field of biomedical research [30].

The cloud-based ecosystem in biomedicine provides a range of benefits and opportunities for researchers. It allows institutions to create a customizable digital infrastructure that supports collaboration and open science. The principles of Findability, Accessibility, Interoperability, and Reusability (FAIR) guide data management and enhance the reusability of data, algorithms, tools, and workflows. Biomedical data ecosystems should be able to include indexing capabilities and align with initiatives to facilitate efficient search and retrieval.

Linux container technologies like Docker and workflow languages such as WDL enable the packaging and orchestration of complex software tools, while cloud providers offer batch-processing capabilities for large-scale data analysis. The core of a cloud-based platform is a shared commons ecosystem, where computing capacity, storage resources, databases, and informatics tools are co-located for easy access and collaboration. Interoperability between multiple commons is facilitated by digital IDs, metadata, APIs, data portability, data peering, and pay-for-compute mechanisms. Features like indexing and search capabilities, meta-learning frameworks for algorithm selection, and integration of established bioinformatics software tools enhance the functionality of the ecosystem.

Security is paramount in a cloud commons architecture, with robust controls, risk assessments, and regular training programs, data users can protect sensitive data and minimize vulnerabilities. Moreover, cryptographic techniques increase security in data outsourcing and collaboration in the cloud. Through these technologies, the cloud-based ecosystem supports open biomedical data access while maintaining security and privacy.

One example of the cloud-based ecosystem is Multi-X [13], which is a research-oriented platform designed for collaborative and reproducible science. It provides a scalable and integrative computational framework that fosters the development and integration of scientific tools. The platform comprises six key components: DATA, for flexible data repositories; ANALISE, for multi-domain scientific tools; COMPUTE, for on-demand computational resources; WORKFLOW, for automated data analytics processes; EXPLORE, for analysis dashboards and visualizations; and COLLABORATE, for web-

based interfaces and sharing capabilities. Through these components, MULTI-X enables researchers to connect with large-scale data repositories, gain immediate insights, and streamline collaboration within and across different domains. It offers a comprehensive ecosystem for cross-domain research, enhancing reproducibility and facilitating scientific exploration.

By embracing the cloud-based ecosystem, researchers can benefit from interoperability, scalability, and flexibility, driving advancements in biomedical research and fostering open science initiatives.

4 Big Data and Cloud in Healthcare

Among the whole medical industry, healthcare attaches more importance to diagnosing, preventing illnesses, and maintaining physical, mental, and emotional well-being. Considering its extensive composition, it is also a highly information-intensive industry, characterized by a vast and constantly updated collection of specialized data from diverse sources[18]. And since the last decade of the twentieth century, with the digitization of health data and the tendency of adopting electronic health records(EHR)[38], the healthcare field has gained significant advances and breakthroughs by utilizing both the collected digital health data and big data and cloud technologies[27]. This section will explore the impact of big data and the cloud on the healthcare industry such as healthcare delivery and personal health management. It will discuss their applications, advances, as well as the existing inadequacies and challenges within the healthcare field.

4.1 Healthcare Big Data

Healthcare data is composed of a diverse range of information, which can be categorized into two main types: biomedical data and non-medical data. As it has been mentioned in the last section, biomedical data includes medical imaging data, EHR data(e.g. medical history, diagnoses, medications and etc), public health surveillance data and etc. Making the most of biomedical data, there is a significant potential to enhance the diagnosis and prevention of diseases as well as foster healthy habits and practices[28]. Biomedically unrelated data in healthcare refers to information that is not directly related to the medical or clinical aspects but still holds relevance and impact on healthcare. Such as the web and social media data from patients, publications(i.e. health reference materials and clinical research reports), patients' feedback, and other important non-medical data[39]. These sources provide valuable insights into patient experiences, healthcare trends, public health issues, and other factors that can influence healthcare decision-making, research, and policy development.

Healthcare data qualifies as big data due to several key factors. First, it involves large volumes of datasets, namely there is a huge amount of data involved which is also increasing at a high speed. Second, the high velocity. That is to say, the speed of continuously generating new data and processing data for clinical decisions is very high. Finally, healthcare data also shows a great variety which contains various formats and broad resources. Specifically, there are mainly three kinds of formats, which are structured data, semi-structured data, and unstructured data. Structured data is stored in a predefined format and it's specific, such as clinical or health data, it's easy to store, manipulate and analyze. Unstructured data is stored in the original format without a predefined manner while semi-structured is between these two formats with partially consistent characteristics in format. Almost healthcare data is saved as unstructured or semi-structured, such as doctor notes, office medical records, and images[33].

Hence, with the integration of healthcare big data and advanced big data analysis techniques, along with cloud computing, we shall present a more personalized and efficient healthcare industry.

4.2 Accurate analysis and prediction in healthcare

As it's mentioned in the last section of the biomedical part, machine learning, such as deep learning algorithms can be used in comparing new patients with large populations to support risk assessment, treatment, and other critical clinical decisions. Considering the characteristic of healthcare big data as complex sources with practical applications, achieving high accuracy in analysis and prediction shall be more important in the healthcare industry, especially from the user's perspective. The reliability and precision of these analyses and predictions play a vital role in delivering effective and personalized healthcare services. Therefore, this part shall focus more on accuracy improvement to improve the quality of decision-making.

Machine learning focuses on developing algorithms and models that can learn from and make predictions or decisions based on the processed data. To improve the accuracy of disease prediction, the most intuitive and effective way is to get high-quality data with complete and accurate information. The source of healthcare data is various and complex, it can be the data from patients' descriptions or doctors' notes, it can also be biomedical data from medical devices, such as blood pressure, temperature, heart rate, etc. However, most of the raw data of healthcare is incomplete with missing values, or vague with inaccurate and even incorrect descriptions. To address the challenges of undesired data, there are two kinds of strategies, regarding the data source and data process respectively.

From the perspective of the data source, accurate healthcare data can be obtained directly from medical sensors by combining the healthcare system with the Internet of Things (IoT). The IoT refers to a network of embedded items, such as physical devices, vehicles, home appliances, etc, that are equipped with electronics, software, sensors, actuators, and connectivity. This allows these objects to connect with each other and exchange data[35]. So for the healthcare of medical usage, the development of wireless sensors and wearable technology has made it possible to monitor a patient's multiple vital signs anywhere and anytime. By utilizing wireless sensors and wearable devices, healthcare providers can continuously monitor crucial parameters with accurate and real-time values(e.g. heart rate). One important application of healthcare big data combining the IoT is the medical body area networks (MBANs)[1], equipped with medical sensors, MBANs can continually monitor the patients' condition by sensing and transmitting the body measurements, like the body temperature, blood pressure, respiratory rate, blood glucose, oxygen saturation and, etc. This data can provide an accurate picture of a patient's physical condition, which is invaluable for both diagnosis and prediction through machine learning, as opposed to the vague descriptions that some patients give of their own bodies.

From the perspective of the data process, there are still lots of methods to estimate and reconstruct the incomplete data. One of a powerful way is to use the latent factor model[6]. This model serves as a feature extraction tool to capture the relationship between observable variables and latent variables[4]. The observed variables refer to those variables that can be directly measured and quantified while the latent variables are measured indirectly through the relationships and patterns observed in the observed variables, rather than measuring things that can't be quantified. In order to reconstruct missing values, the latent factors are able to capture the underlying structures and relationships within the observed data. This enables estimating and filling in the missing values based on the information captured by the latent factors so as to get the complete information.

4.3 Cloud system for healthcare big data

The rapid development in information technology indeed has brought significant progress to the healthcare industry across different domains, like using healthcare big data to analyze and predict diseases combing with machine learning and IoT as mentioned above. However, these advancements have also resulted in an exponential increase in the volume and complexity of healthcare data, making it increasingly challenging to handle and process effectively. Like the data collected from different sensors in the MBANs, it's with a huge amount that can be generated within a short time. Additionally, it's also heterogeneous as there are different kinds of formats(e.g. text, image, etc.), which needs to be processed in advance. Hence, the high volume, velocity, and variety of healthcare big data contribute to the complexities of processing, managing, and extracting valuable insights from those data. Dealing with such data requires specialized techniques and technologies capable of handling the scale and diversity of information effectively.

As cloud computing is a technology that is used for effective virtualization, process power, storage, connectivity, and resource pooling[16]. Specifically, the process power refers to advantage of cost-effectiveness in terms of both energy and financial resources during data processing in a cloud-based environment, while the connectivity enable data from different sources shall relate to each other and the users within the cloud services are able to interact with the Internet. Regarding these features, the introduction of the cloud allows these resources to be accessed and shared across multiple devices effectively through a wide network, such as the Internet. With a cloud computing infrastructure merging into the healthcare industry, patients can obtain services on demand, such as real-time monitoring of the medical signs at home, information island, teleradiology, and telemedicine systems, etc. In the meantime ensuring key concepts like isolation, security, distribution, and resilience are upheld. Hence, cloud computing and cloud-based system perfectly matches the needs of healthcare big

data, namely integrating resources on a cloud platform that combines with state-of-the-art technology shall solve most of the issues the healthcare big data faces, such as processing heterogeneous data, data management, etc.

Such as the Healthcare Cyber-Physical System(Health-CPS) based on cloud and big data[42]. It's a cloud-based and patient-centric system of three layers. Data collection layer, data management layer, and application service layer. The data collection layer, it's used to collect raw data from different sources(e.g. clinics, research institutes, etc) from a data node. Then a built-in component named adaptor shall preprocess data with a unified standard, and encrypt preprocessed data to ensure the security of the collected data. The preprocessed and encrypted data includes data description, data entity, and security tag. After that, it will get access to the data management layer, which includes two modules, a distributed file storage (DFS) module and a distributed parallel computing (DPC) module. The DFS module will enhance the performance of healthcare data with efficient storage, high throughput, and high fault tolerance. The DPC supports both real-time analyses as well as offline analysis, combing with some data mining and recommendations algorithms. Depend on different situations, the focus shall also be different. The real-time analysis requires quick responses while the offline focus more on personalization. And the last is the application service layer for practical usages, such as the monitor-based application equipped with sensors, it can not only offer real-time sensing and transmit some vital body measurements to prevent an emergency but also enable offline analysis for long time conditioning.

5 Discussion

In the preceding sections, we provided detailed explanations of the utilization of big data in the biomedical and healthcare domains, as well as the adoption of cloud-based solutions to address associated challenges. While big data and cloud technologies have undoubtedly contributed to substantial advancements in the medical sector, it is imperative to recognize the existence of certain limitations and challenges. Given the accelerating trend towards patient-centric treatment and healthcare, cloud platforms and services have introduced security measures to authenticate data access and safeguard sensitive information. However, it is crucial to acknowledge that security and privacy concerns have not been entirely overcome.

Regarding the issue of security, the prevailing trend in healthcare systems is the adoption of cloud-based infrastructures. However, it is crucial to acknowledge the vulnerability of the cloud to cyber attacks. Since all user information is stored in the cloud, any breach in the cloud systems poses a significant risk of data leakage. Given the sensitive nature of the user data involved, such a vulnerability would have severe consequences for user privacy and the reputations of healthcare institutions. The potential harm resulting from the exposure of user data is, therefore, a matter of great concern. One potential solution to mitigate the potential loss of critical data is the implementation of the decoy technique [36]. This approach involves the retrieval of decoy files when an attacker is detected accessing the system data, thereby enhancing data security. Furthermore, to enhance the security level, the original files can be automatically encrypted once attacks are detected. However, despite the cloud providing a barrier to user access and the application of security techniques to defend against attackers, there is no guarantee of complete data protection.

In terms of privacy, clinical records and healthcare data are obtained with the consent of patients and are safeguarded by medical institutions and centers. As a result, organizations, research teams, and analytical firms are required to obtain patient consent before utilizing any datasets in order to mitigate the risks of data exploitation and identity misuse. It is crucial to acknowledge that the outcomes and clinical characteristics of patients involved in studies become publicly accessible, thereby increasing the potential for breaches in privacy and improper utilization of patient data. Consequently, the issue of accessing patient data from publicly available sources remains a highly sensitive concern in the healthcare domain, necessitating stringent measures to protect patient privacy and ensure responsible data usage [15]. For this issue, there is an urgent need to address the inadequacy of policies related to the utilization of patients' data. It is crucial to establish more comprehensive policies that ensure appropriate and ethical usage.

In the future, several key actions can be implemented to enhance the utilization of big data in the medical industry. These approaches aim to promote collaboration among various entities such as firms, industries, and data-generating teams to ensure equitable access to data while minimizing

the generation of redundant and low-quality data. This collaborative effort seeks to overcome the challenge of data overload and improve the overall efficiency of data analysis. Additionally, the approval of artificial intelligence (AI) and machine learning (ML) modules and methods by regulatory agencies is crucial to facilitate their integration into therapeutic and drug discovery pipelines. This regulatory endorsement will provide a framework for the safe and effective application of AI-ML techniques in medical industries, enabling the development of innovative solutions and advancements in medical research. It is also important to address compliance issues that arise throughout the process of collecting, managing, and analyzing medical big data. All relevant stakeholders must be mindful of their compliance obligations in accordance with the specific requirements of their respective fields. This includes adhering to ethical guidelines, privacy regulations, and data protection laws to ensure the responsible and lawful handling of medical big data [17] [8].

6 Conclusion

In summary, big data and cloud technologies play crucial roles in the modern medical industry. The utilization of these technologies gives rise to a variety of novel applications and opportunities, which are capable of enhancing various aspects within the medical industry. Nevertheless, a series of challenges also emerge alongside this evolution. Specifically, this paper elaborates on two segments within the general medical industry, namely biomedicine and healthcare, to provide more contextual and intuitive insights into the comprehension of the research question: how do big data and cloud impact the modern medical industry? Our investigation into the literature shows that big data technologies have the potential to revolutionize the industry by providing valuable insights, mitigating health risks, minimizing harmful environmental exposures, improving patient care, and enhancing overall health outcomes [8], while cloud technologies offer opportunities to deploy and integrate big data applications with better usability, accessibility, interoperability, scalability, cost efficiency, and flexibility. However, concerns with these technologies regarding security and privacy are not yet entirely conquered. Through a thorough discussion section, this paper also exhibits how researchers and developers are actively coping with these challenges from both technology and political perspectives. Consequently, by conducting this literature review, we wish to provide a comprehensive and coherent analysis which helps researchers to better understand and study the concepts, applications, and challenges of big data and cloud technologies in the medical industry.

Work Distribution

Table 1: Work Distribution

Name	Work
Yudong Fan	Abstract, Section 1,2,6
Yuna Chen	Section 3, 5
Shutong Cai	Section 4, 5

References

[1] Asif Adil, Hushmat Amin Kar, Rajendra Jangir, and Shabir Ahmad Sofi. Analysis of multi-diseases using big data for improvement in healthcare. In *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, pages 1–6, 2015.

[2] Enis Afgan, Brad Chapman, and James Taylor. Cloudman as a platform for tool, data, and analysis distribution. *BMC bioinformatics*, 13:1–7, 2012.

[3] Giuseppe Agapito, Barbara Calabrese, Pietro H Guzzi, Gionata Fragomeni, Giuseppe Tradigo, Pierangelo Veltri, and Mario Cannataro. Parallel and cloud-based analysis of omics data: Modelling and simulation in medicine. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 519–526. IEEE, 2017.

[4] Li Cai. Latent variable modeling. *Shanghai archives of psychiatry*, 24:118–20, 04 2012.

[5] Barbara Calabrese and Mario Cannataro. Cloud computing in healthcare and biomedicine. *Scalable Computing: Practice and Experience*, 16(1):1–18, 2015.

[6] Min Chen, Yixue Hao, Kai Hwang, Lu Wang, and Lin Wang. Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, 5:8869–8879, 2017.

[7] Fabricio F. Costa. Big data in biomedicine. *Drug Discovery Today*, 19(4):433–440, 2014.

[8] Conor John Cremin, Sabyasachi Dash, and Xiaofeng Huang. Big data: Historic advances and emerging trends in biomedical research. *Current Research in Biotechnology*, 4:138–151, 2022.

[9] Vasiliki Danilatou and Sotiris Ioannidis. Security and privacy architectures for biomedical cloud computing. In *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, pages 1–4. IEEE, 2010.

[10] Tuhin Das, Tushar Kanti Das, Anne Khodarkovskaya, and Sabyasachi Dash. Non-coding rnas and their bioengineering applications for neurological diseases. *Bioengineered*, 12(2):11675–11698, 2021. PMID: 34756133.

[11] Sabyasachi Dash, Sushil Kumar Shakyawar, Mohit Sharma, and Sandeep Kaushik. Big data in healthcare: management, analysis and future prospects. *Journal of Big Data*, 6(1):1–25, 2019.

[12] Andrea De Mauro, Marco Greco, and Michele Grimaldi. What is big data? a consensual definition and a review of key research topics. In *AIP conference proceedings*, volume 1644, pages 97–104. American Institute of Physics, 2015.

[13] Milton Hoz de Vila, Rahman Attar, Marco Pereanez, and Alejandro F Frangi. Multi-x, a state-of-the-art cloud-based ecosystem for biomedical research. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1726–1733. IEEE, 2018.

[14] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[15] Khaled El Emam, Sam Rodgers, and Bradley Malin. Anonymising and sharing individual patient data. *bmj*, 350, 2015.

[16] Faraz Fatemi Moghaddam, Touraj Khodadadi, Kasra Madadipouya, Mohammad Ahmadi, and Mahsa Rohani. Cloud computing: Vision, architecture and characteristics. 08 2015.

[17] Xunjie Gou and Zeshui Xu. An overview of big data in healthcare: multiple angle analyses. *Journal of Smart Environments and Green Computing*, 1(3):131–145, 2021.

[18] Ishita Goyal, Amritanshu Singh, and Jaspal Kaur Saini. Big data in healthcare: A review. In *2022 1st International Conference on Informatics (ICI)*, pages 232–234, 2022.

[19] Stefan Graw, Kevin Chappell, Charity L Washam, Allen Gies, Jordan Bird, Michael S Robeson, and Stephanie D Byrum. Multi-omics data integration considerations and study design for biological systems and disease. *Molecular omics*, 17(2):170–185, 2021.

[20] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, 47:98–115, 2015.

[21] Allison P Heath, Matthew Greenway, Raymond Powell, Jonathan Spring, Rafael Suarez, David Hanley, Chai Bandlamudi, Megan E McNerney, Kevin P White, and Robert L Grossman. Bionimbus: a cloud for managing, analyzing and sharing large genomics datasets. *Journal of the American Medical Informatics Association*, 21(6):969–975, 2014.

[22] Laurent Jourdren, Maria Bernard, Marie-Agnès Dillies, and Stéphane Le Crom. Eoulsan: a cloud computing-based framework facilitating high throughput sequencing analyses. *Bioinformatics*, 28(11):1542–1543, 2012.

[23] Johan Karlsson, Oscar Torreno, Daniel Ramet, Günter Klambauer, Miriam Cano, and Oswaldo Trelles. Enabling large-scale bioinformatics data analysis with cloud computing. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, pages 640–645. IEEE, 2012.

[24] Ava L Liberman and David E Newman-Toker. Symptom-disease pair analysis of diagnostic error (spade): a conceptual framework and methodological approach for unearthing misdiagnosis-related harms using big data. *BMJ quality & safety*, 27(7):557–566, 2018.

[25] Chih-Wei Lu, Chih-Ming Hsieh, Chih-Hung Chang, and Chao-Tung Yang. An improvement to data service in cloud computing with content sensitive transaction analysis and adaptation. In *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, pages 463–468. IEEE, 2013.

[26] Jake Luo, Min Wu, Deepika Gopukumar, and Yiqing Zhao. Big data application in biomedical research and health care: a literature review. *Biomedical informatics insights*, 8:BII–S31559, 2016.

[27] D. Mendelson and D. Mendelson. Legal protections for personal health information in the age of big data – a proposal for regulatory framework. *Ethics, Medicine and Public Health*, 3(1):37–55, 2017.

[28] Brent Daniel Mittelstadt and Luciano Floridi. The ethics of big data: Current and foreseeable issues in biomedical contexts. *Science and Engineering Ethics*, 22(2):303–341, 2016.

[29] Thilo Muth, Julian Peters, Jonathan Blackburn, Erdmann Rapp, and Lennart Martens. Proteocloud: A full-featured open source proteomics cloud computing pipeline. *Journal of Proteomics*, 88:104–108, 2013.

[30] Vivek Navale and Philip E Bourne. Cloud computing applications for biomedical science: A perspective. *PLoS computational biology*, 14(6):e1006144, 2018.

[31] Devansh Patel, Dhwanil Shah, and Manan Shah. The intertwine of brain and body: a quantitative analysis on how big data influences the system of sports. *Annals of Data Science*, 7:1–16, 2020.

[32] Wahid Rajeh. Hadoop distributed file system security challenges and examination of unauthorized access issue. *Journal of Information Security*, 13(2):23–42, 2022.

[33] Satwik Sabharwal, Samridhi Gupta, and K. Thirunavukkarasu. Insight of big data analytics in healthcare industry. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 95–100, 2016.

[34] Michael C Schatz. Cloudburst: highly sensitive read mapping with mapreduce. *Bioinformatics*, 25(11):1363–1369, 2009.

[35] Muhammad Shafiq, Zhaoquan Gu, Omar Cheikhrouhou, Wajdi Alhakami, and Habib Hamam. The rise of "internet of things" review and open research issues related to detection and prevention of iot-based security attacks. *Wireless Communications and Mobile Computing*, 2022:12, 08 2022.

[36] E Shanmugapriya and R Kavitha. Medical big data analysis: preserving security and privacy with hybrid cloud technology. *Soft Computing*, 23:2585–2596, 2019.

[37] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[38] Chandrasekar Vuppalapati, Anitha Ilapakurti, and Santosh Kedari. The role of big data in creating sense ehr, an integrated approach to create next generation mobile sensor and wearable data driven electronic health record (ehr). In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 293–296, 2016.

[39] Lidong Wang and Cheryl Ann Alexander. Big data analytics in medical engineering and healthcare: methods, advances and challenges. *Journal of Medical Engineering & Technology*, 44:267 – 283, 2020.

[40] Chaowei Yang, Qunying Huang, Zhenlong Li, Kai Liu, and Fei Hu. Big data and cloud computing: innovation opportunities and challenges. *International Journal of Digital Earth*, 10(1):13–53, 2017.

[41] Tianyi Yang and Yang Zhao. Application of cloud computing in biomedicine big data analysis cloud computing in big data. In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pages 1–3. IEEE, 2017.

[42] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95, 2017.

Convergence of IoT, Edge Computing, and Cloud Computing: An Overview and Comparative Analysis

Boyuan Xiao
boyuan.xiao@student.uva.nl
Sanskar Bajpai
sanskar.bajpai@student.uva.nl
Yufei Wang
yufei.wang2@student.uva.nl

May 30, 2023

Abstract

The convergence of Internet of Things (IoT), Edge Computing, and Cloud Computing has gained significant attention in recent years due to its high performance and cost efficiency. In this paper, we provide an overview and key components of these technologies, exploring their benefits and challenges. We discuss how the combination of edge computing's low latency and high security, cloud computing's scalability and flexibility, and IoT devices' ability to connect with the physical world can revolutionize the capabilities of IoT devices. This paper also includes real-world examples to illustrate how these technologies are being effectively utilized together in various domains. Furthermore, the paper explores challenges and future trends in the industry and offers observations on the potential impact of this convergence across various domains.

1 Introduction

In the past few years, smart devices, such as smartphones and smartwatches, gained a huge user base. Along with the advent of 4G and 5G networks, Edge Computing, which benefits a lot from the rise of these two technology, becomes popular in the industry. As the name suggests, Edge Computing mainly happens at the edge of the internet and processes various data in a closer geographical location to the user. Therefore, it is obvious that one of the biggest advantages of Edge Computing is low-latency response[1]. In the meantime, Cloud Computing thrives and changed our lifestyles significantly as well. As the era of Cloud Computing 3.0 approaches, the enterprise IT's own application architecture has gradually shifted from (relying on traditional business databases and middleware

business suites, specifically designed for each business application field, chimney-like, high-complexity, stateful, large-scale) vertical scale application layered architecture to (relying on open source enhanced, highly shared across different business application domains) database, middleware platform service layer and (more lightweight and decoupling functions, complete separation of data and application logic) distributed stateless architecture[2]. This shift of focus, in turn, enables a better interplay between Edge Computing and Cloud Computing.

Next to Edge Computing, another frequently mentioned concept is the Internet of Things (IoT). The number of IoT devices that are active grows at an incredible speed in recent years, and this number is expected to hit 50 billion in 2025[3]. Moreover, the relationship between Edge Computing and IoT is convoluted. IoT enables multiple new technologies while influencing existing research fields, such as Machine Learning[4] and Deep Reinforcement Learning[5].

This literature review aims to discuss the current development and convergence of Cloud Computing, Edge Computing and IoT. Sections 2, 3 and 4 present a comprehensive view of Edge Computing, Cloud Computing and IoT respectively. Section 5 gives a closer look of the relationship between Cloud Computing and Edge Computing while Section 6 focuses on the convergence of all three. Some of the existing obstacles faced by all three of them are listed in Section 7. Finally, we show our observation of the future trends of the industry and conclusion in Sections 8 and 9.

2 Edge Computing

2.1 Overview

As previously mentioned in Section 1, Edge Computing sits at the edge of the network. It brings data processing and computation closer to the source of data generation, rather than relying on a centralized cloud infrastructure. This distributed computing paradigm benefits from the fact that the computation power, namely the CPU, of edge devices grows significantly over the past few years. Edge Computing has been acknowledged to overcome the issues caused by the conventional centralized computation network formed by cloud computing alone, e.g., single point of failure and excessive traffic/computation burden caused by data aggregation[6]. It is common that people often mention IoT along with Edge Computing. Edge-Computing-Driven IoT (ECDriven-IoT) is strong proof for the fact that they are closely related. ECDriven-IoT is a promising solution taking advantage of edge computing to build scalable and efficient IoT systems. As a combination of the two, ECDriven-IoT managed to overcome issues, such as how to achieve good interconnection and interoperability among IoT devices, for traditional IoT with the help of Edge Computing[1].

2.2 Key Components

Edge devices, which are the physical component of Edge Computing, is crucial. Meanwhile, since the network architecture becomes rather distributed, it is really essential that the network connections between edge devices have low latency. The goal of such an architecture is to perform the delay-sensitive and computationally intensive part of an application in the edge network[7]. As a result, **low-latency network** plays an important role in Edge Computing.

3 Cloud Computing

3.1 Overview

Different from both Edge Computing and IoT, Cloud Computing is a well-known concept with a history of longer than two decades[8]. It allows users to use IT services as easily as using water from a water faucet. More importantly, users can use them without worrying about purchasing the underlying hardware or coping with the complex setup. Some of the main properties of Cloud Computing are On-Demand Self-Service, Extensive Network Access, Resource Pooling, Fast and Elastic Scaling and Measurable Services[2]. After more research and practice, scholars believe that it is necessary to decentralize Cloud Computing. Boundaries between Edge Computing and Cloud Computing blur as technologies like Ad-hoc Edge Cloud architectures and Swarm computing emerge[9].

3.2 Key Components

There are many fundamental technologies which enable Cloud Computing. The importance of **Broadband Network and Internet Architecture** goes without saying. **Data Center Technology**, **Virtualization Technology** and **Multi-Tenant Technology** defines the architecture of Cloud Computing. **Web Technology** is often used as the realization medium and management interface of cloud services[2].

4 Internet of Things (IoT)

4.1 Overview

Internet of Things was first introduced by Marc Weiser in 1991 in his book "The Computer for the Twenty-first Century". IoT technology benefits a wide range of fields including agriculture, healthcare, transportation & logistics and smart cities. The definition of IoT also varies in different scientific fields. However, the book[10] gives a comprehensive definition of IoT devices covering many aspects. When looking at the hardware definition of IoT, sensors and actuators make IoT distinguishable from others. By using sensors and actuators, IoT devices manage to extract, pre-process, interpret and transfer data to other devices or other receiving entities in the Internet.

4.2 Key Components

Except for **sensors and actuators**, IoT requires **internet connectivity** similar to all the other internet-based technologies. On the other side of the network connection, **cloud infrastructures** frequently communicate with IoT devices. Cloud Computing plays a significant role in IoT deployments. Cloud infrastructure provides storage, computing power, and data processing capabilities necessary for handling large volumes of IoT data. It enables scalable and flexible data storage, analytics, and access to services and applications.

5 Relationship between Edge Computing and Cloud Computing

To begin with, we take a look at the differences between these two computing technologies. The crucial difference is that they process data at different locations, which also results in their different features [11]. Edge computing processes data closer to the data source. It brings computing resources and data storage closer to the devices generating the data, such as IoT devices, sensors, or mobile devices. Cloud computing, on the other hand, centralizes data processing and storage in remote data centres or the cloud. It relies on a network of powerful servers and infrastructure to handle data processing, storage, and computing tasks. Users can only access and utilize cloud resources over the internet. Thus it provides the different properties as follows.

Latency: edge computing excels in scenarios where low latency and real-time processing are critical. By processing data locally at the edge, edge computing can provide faster response times and immediate decision-making. This is particularly advantageous for applications requiring real-time analysis, such as autonomous driving, and remote monitoring. While cloud computing can handle real-time tasks to some extent, it is not an ideal choice for applications that require immediate responses. This is because the need for internet connection and data transmission to the cloud introduces delays and latency, which makes it less suitable for such time-sensitive applications. As Wan [12] analyzes in the paper, the current distributed Cloud introduces unpredictable latencies caused by dispatching, network transmission and computation compared with the traditional computing model. It challenges Cloud Computing and hinders the effort to migrate applications to the Cloud. Liu et al. [13] present the opportunities and challenges of applying edge computing for automatic driving. As the vehicle constantly collects data from various sensors, such as cameras, lidar, and radar, edge computing is preferred over cloud computing since it allows for immediate data processing and decision-making directly within the vehicle. By processing the data locally, edge computing reduces the reliance on the cloud and minimizes the latency introduced by transmitting data to a remote server. Otherwise, the vehicle may react slowly and passengers can face dangers because of the low efficiency of data flow and analysis.

Scalability: cloud computing is highly scalable. It can easily R/W comput-

ing resources and dynamically allocate computing resources. Edge computing, however, typically operates on a smaller scale and involves localized resources. The computing capabilities at the edge are limited by the physical hardware and network infrastructure. Because cloud computing providers maintain large data centres with extensive server infrastructure and storage capacity, applications on the cloud are able to scale and handle increased workloads without limitations imposed by local edge devices. Therefore, cloud computing is an essential part of the big data field [14]. Various cloud performance benchmarks and evaluations [15] [16] demonstrated that balancing the number and size of VMs as a function of the specific applications is critical to achieving optimal scalability for geospatial big data applications. It provides effective tools to manage, process, and analyze for big data.

Security: edge computing performs better on data privacy and security since sensitive data can be processed and analyzed locally without being transmitted to the remote server. This can be crucial for industries with strict data privacy regulations or applications dealing with sensitive user information. In Carlin and Curran's paper [17], they illustrate the main security risks and issues that are currently present within the cloud computing industry. Although cloud computing applies robust security measures implemented by cloud service to protect data, there are still concerns over the data privacy caused by compliance issues or authorized access.

In summary, even if edge and cloud computing are both state-of-the-art computing methods nowadays, they are not perfect and have some defects in some application scenarios. While edge computing enables real-time processing, reduces latency, and protects data privacy, cloud computing provides scalable infrastructure and better capabilities to handle large-scale processing. Therefore, a hybrid approach combining both computing technologies can be adopted to improve the performance of systems.

6 Convergence

As discussed above, edge computing and cloud computing have their pros and cons. Thus how to make use of their advantages and combine them with IoT becomes a transformative trend. The convergence is revolutionizing the way we interact with technology and the capabilities of IoT devices. It brings the low latency and high security of edge computing, scalability and flexibility of cloud computing, as well as the ability of IoT devices to connect with the physical world. We analyze how IoT benefits from edge and cloud computing with instances.

6.1 Real-time Solution

Edge computing enables rapid analysis and decision-making at the edge devices, reducing latency and enabling real-time responses. Sood et al. [18] developed an Edge Cloud-centric IoT-based smart traffic management system for traffic inflow prediction and time-optimized smart navigation of vehicles. By accurately

predicting traffic movement, the system can adjust traffic signal timings at intersections and prevent long waiting queues and congestion. Additionally, through smart navigation, the system can distribute traffic optimally across different paths, improving road safety at intersections. This system leverages real-time data from connected devices, processes it locally with edge computing, and utilizes cloud resources for complex analytics and decision-making. Therefore, critical data can be processed locally, while non-time-sensitive data can be sent to the cloud for further analysis, which makes provide real-time robust and smart solutions to IoT systems.

6.2 Big Data Analysis

As cloud computing is widely used in IoT and big data field, IoT Cloud becomes a term that represents the technology architecture that connects IoT devices to servers housed in cloud data centres. While the Internet of Things refers to the world's collection of devices that gather and share information across various industries and sectors, cloud computing offers storage and processing capabilities for huge amounts of data among clusters. IoT Cloud is affecting our life in every corner. Take healthcare for example, IoT Cloud is revolutionizing eHealth and its whole ecosystem. Nowadays, IoT devices are indispensable in medical systems. With combination with cloud computing, the systems are able to store and process the immense volume of data generated by IoT devices. According to the models learn from vast amounts of previous patients' data, patients can get detailed reports and even personalized healthcare. Aceto et al. [19] discussed the relationship between health and these technologies, believing IoT, cloud computing, and big data are moving it towards HealthCare 4.0. Besides the medical field, cloud computing enables IoT deployments to grow rapidly and adapt to changing requirements in various areas.

6.3 Data Privacy

IoT devices are not usually built with security solutions, leading to potential vulnerabilities in a multiple-device system. Edge computing, however, can help reduce these risks. It provides encrypted tunnels and access control to secure access. Besides, edge computing is not centralized, it is able to implement threat detection technologies to identify a potential breach as early as possible. Security agents or micro data centres, for example, can be applied on edge nodes to prevent unauthorized or malicious traffic from compromised IoT devices. According to the levels of data sensitivity, systems are supposed to store and process different data between edge and cloud computing, which will greatly enhance data privacy issues.

6.4 Cost Efficiency

The distribution between edge and cloud computing can optimize the costs. First, with edge computing, systems can send only relevant and delayed data to

the cloud, which will significantly reduce bandwidth requirements and associated costs. It is particularly important in scenarios where network connectivity is limited or expensive. Additionally, cloud computing offers elasticity, allowing it to dynamically scale the computing resources based on demand. Some dynamic task allocation algorithms were also studied by Ding et al. [20] and Du et al. [21] to make the best use of the available resources.

Overall, the convergence of IoT, edge computing, and cloud computing optimizes how data is stored, processed, and managed. The convergence utilizes the strengths of different technologies and creates a powerful and complementary ecosystem. It opens up new possibilities and opportunities across various industries, ranging from traffic and healthcare to smart cities.

7 Challenges

7.1 Challenges of IoT

Rob et al. [22] states challenges regarding IoT devices, some the highlights are:

Zero-Entropy systems: Energy will be a major technological challenge in the next five to 10 years, and research must be conducted in order to develop systems that are able to harvest energy from the environment and not waste any under operation.

Security & Privacy: It is crucial to effectively tackle and resolve the challenge of ensuring adequate security on devices that have limited capabilities. Additionally, there is a need to establish and implement technological frameworks prioritising privacy protection, serving as the foundation for future advancements.

Scalability: The Internet of Things (IoT) is anticipated to consist of an immense number of devices, potentially reaching trillions. Although it is improbable for all devices to be interconnected in a mesh-like structure, they will likely be organized into hierarchical subdomains. Nonetheless, the scale of interconnected objects within the IoT is expected to surpass the current internet by several orders of magnitude.

7.2 Challenges of Edge Computing

Varghese et al. [23] discuss the following challenges:

Heterogeneity: The edge computing landscape is highly heterogeneous, with diverse devices, platforms, and network technologies. Achieving interoperability, seamless integration, and efficient communication across this heterogeneous environment is challenging.

Data Management and Analytics: Managing and processing data at the edge efficiently is crucial for real-time decision-making. Edge analytics, data aggregation, filtering, and synchronization techniques need to be developed to handle data in a distributed and dynamic environment.

Quality of Service: Meeting the desired quality of service requirements in terms of latency, reliability, and availability is a challenge in edge computing.

Efficient task scheduling, service placement, and network optimization techniques are needed to deliver satisfactory performance.

Resource Constraints: Resource Constraints: Edge devices typically have limited resources such as processing power, memory, and energy. Optimizing resource utilization and designing lightweight algorithms and protocols are essential to operate within these constraints.

7.3 Challenges of Cloud Computing in the Context of IoT Devices

Sadeeg et al. [24] include the following challenges:

Storage and Computational performance: Systems that include the use of cloud-based IoT devices require a high degree of performance goal requirements. Such specifications can be difficult to meet in all settings because cloud-based IoT devices are in motion for many applications

Reliablitiy: IoT devices are dependent on the Cloud to work providers for time-critical apps, and the effect would directly reflect the program's output. In cars, surgical instruments, or in the security field, for example.

8 Trends

Carvalho et al. [25] states the following: The rapid growth of mobile devices connecting to the network edge presents both opportunities and challenges. While mobility enhances flexibility for users and applications, it also leads to frequent disconnections between edge devices and the network, negatively impacting service quality in terms of loss, delay, and bandwidth.

However, there is a need for a comprehensive management architecture that facilitates seamless device and service handover, along with fault-tolerance systems to mitigate failures. Considering different network access technologies and administrative domains, optimization of both horizontal and vertical handovers is crucial. This optimization should account for factors beyond signal quality, including movement direction, network cost-benefit analysis, and service quality.

Furthermore, the mobility of edge nodes intensifies requirements for resource availability, resource discovery, task offloading, and resource provisioning [125]. User mobility also affects the number of hops between users and their services, particularly at network boundaries. Therefore, efficient and dynamic migration or replication of edge services becomes imperative in response to such network transitions.

Edge computing can leverage the cloud to enhance reliability. By storing data and running applications on Cloud servers, the risk of data and application loss on mobile devices is reduced. However, it is crucial to establish dependable edge systems that do not rely on Cloud servers. Additionally, addressing challenges such as individual device failures, network coverage issues, network failures, platform failures, and user interface failures requires the development of a reliable and fault-tolerant edge environment.

9 Conclusion

The convergence of IoT, Edge Computing, and Cloud Computing has paved the way for transformative advancements in the field of technology. This convergence offers numerous opportunities to create intelligent, efficient, and interconnected systems that can revolutionize industries and improve the quality of life for individuals.

By leveraging the capabilities of IoT devices, data can be collected and processed at the network edge through Edge Computing. This allows for real-time analytics, reduced latency, and efficient resource utilization. The Edge Computing layer complements the Cloud Computing infrastructure, which provides scalable storage, computation, and services.

Together, IoT, Edge Computing, and Cloud Computing enable a distributed architecture that offers several benefits. It enables efficient data processing and analysis closer to the source, reducing the need for transmitting large amounts of data to the cloud. This reduces network congestion, minimizes latency, and enhances overall system performance.

Moreover, this convergence enables the seamless integration of various devices, sensors, and applications, fostering the development of innovative solutions across industries such as healthcare, transportation, manufacturing, and smart cities. It facilitates real-time decision-making, predictive analytics, and automation, unlocking new possibilities for optimization, cost reduction, and improved operational efficiency.

In conclusion, the convergence of IoT, Edge Computing, and Cloud Computing has its set of challenges but holds immense potential to drive innovation and transformation across various domains. By effectively harnessing the capabilities of these technologies, organizations can unlock new opportunities and deliver enhanced services, ultimately shaping a more connected and intelligent future.

References

[1] Linghe Kong et al. "Edge-Computing-Driven Internet of Things: A Survey". In: *ACM Comput. Surv.* 55.8 (Dec. 2022). ISSN: 0360-0300. DOI: 10.1145/3555308. URL: https://doi-org.vu-nl.idm.oclc.org/10.1145/3555308.

[2] "Introduction to Cloud Computing Computing". In: *Cloud Computing Technology*. Singapore: Springer Nature Singapore, 2023, pp. 1–58. ISBN: 978-981-19-3026-3. DOI: 10.1007/978-981-19-3026-3_1. URL: https://doi.org/10.1007/978-981-19-3026-3_1.

[3] Ericsson. "CEO to shareholders: 50 billion connections 2020". In: *https://www.ericsson.com/en/press-releases/2010/4/ceo-to-shareholders-50-billion-connections-2020* (2010).

[4] Eunice Likotiko, Yuki Matsuda, and Keiichi Yasumoto. "Garbage Content Estimation Using Internet of Things and Machine Learning". In: *IEEE Access* 11 (2023), pp. 13000–13012. DOI: 10.1109/ACCESS.2023.3242547.

[5] Abdeladim Sadiki et al. "Deep reinforcement learning for the computation offloading in MIMO-based Edge Computing". In: *Ad Hoc Networks* 141 (2023), p. 103080. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2022.103080. URL: https://www.sciencedirect.com/science/article/pii/S1570870522002529.

[6] Hao Ran Chi. "Editorial: Edge Computing for the Internet of Things". In: *Journal of Sensor and Actuator Networks* 12.1 (2023). ISSN: 2224-2708. DOI: 10.3390/jsan12010017. URL: https://www.mdpi.com/2224-2708/12/1/17.

[7] Patrick McEnroe, Shen Wang, and Madhusanka Liyanage. "A Survey on the Convergence of Edge Computing and AI for UAVs: Opportunities and Challenges". In: *IEEE Internet of Things Journal* 9.17 (2022), pp. 15435–15459. DOI: 10.1109/JIOT.2022.3176400.

[8] Kjell Bratbergsengen. "Cloud Computing in the 1970s: The Discovery of Hash Based Relational Algebra". In: *History of Nordic Computing 3*. Ed. by John Impagliazzo, Per Lundin, and Benkt Wangler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 368–374. ISBN: 978-3-642-23315-9.

[9] Ana Juan Ferrer. "Setting the Scene: Cloud, Edge, Mobile and Ad-hoc Computing Context". In: *Beyond Edge Computing: Swarm Computing and Ad-Hoc Edge Clouds*. Cham: Springer International Publishing, 2023, pp. 13–20. ISBN: 978-3-031-23344-9. DOI: 10.1007/978-3-031-23344-9_2. URL: https://doi.org/10.1007/978-3-031-23344-9_2.

[10] Pascal Hirmer. "Foundations". In: *Model-Based Approaches to the Internet of Things*. Cham: Springer International Publishing, 2023, pp. 7–15. ISBN: 978-3-031-18884-8. DOI: 10.1007/978-3-031-18884-8_2. URL: https://doi.org/10.1007/978-3-031-18884-8_2.

[11] GS Sriram. "Edge computing vs. Cloud computing: an overview of big data challenges and opportunities for large enterprises". In: *International Research Journal of Modernization in Engineering Technology and Science* 4.1 (2022), pp. 1331–1337.

[12] Zhitao Wan. "Cloud Computing infrastructure for latency sensitive applications". In: *2010 IEEE 12th International Conference on Communication Technology*. IEEE. 2010, pp. 1399–1402.

[13] Shaoshan Liu et al. "Edge computing for autonomous driving: Opportunities and challenges". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1697–1716.

[14] Chaowei Yang et al. "Big Data and cloud computing: innovation opportunities and challenges". In: *International Journal of Digital Earth* 10.1 (2017), pp. 13–53.

[15] Amril Nazir et al. "Evaluation of virtual machine scalability on distributed multi/many-core processors for big data analytics". In: *2012 IEEE Conference on Open Systems*. IEEE. 2012, pp. 1–6.

[16] Sebastian Lehrig et al. "CloudStore—towards scalability, elasticity, and efficiency benchmarking and analysis in Cloud computing". In: *Future Generation Computer Systems* 78 (2018), pp. 115–126.

[17] Sean Carlin and Kevin Curran. "Cloud computing security". In: *Pervasive and Ubiquitous Technology Innovations for Ambient Intelligence Environments*. IGI Global, 2013, pp. 12–17.

[18] Sandeep Kumar Sood et al. "Smart vehicular traffic management: An edge cloud centric IoT based framework". In: *Internet of Things* 14 (2021), p. 100140.

[19] Giuseppe Aceto, Valerio Persico, and Antonio Pescapé. "Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0". In: *Journal of Industrial Information Integration* 18 (2020), p. 100129.

[20] Shiyao Ding and Donghui Lin. "Dynamic task allocation for cost-efficient edge cloud computing". In: *2020 IEEE International Conference on Services Computing (SCC)*. IEEE. 2020, pp. 218–225.

[21] Mingzhe Du et al. "Algorithmics of Cost-Driven Computation Offloading in the Edge-Cloud Environment". In: *IEEE Transactions on Computers* 69.10 (2020), pp. 1519–1532. DOI: 10.1109/TC.2020.2976996.

[22] Rob Van Kranenburg and Alex Bassi. "IoT challenges". In: *Communications in Mobile Computing* 1.1 (2012), p. 9.

[23] Blesson Varghese et al. "Challenges and opportunities in edge computing". In: *2016 IEEE international conference on smart cloud (SmartCloud)*. IEEE. 2016, pp. 20–26.

[24] Mohammed Mohammed Sadeeq et al. "IoT and Cloud computing issues, challenges and opportunities: A review". In: *Qubahan Academic Journal* 1.2 (2021), pp. 1–7.

[25] Gonçalo Carvalho et al. “Edge computing: current trends, research challenges and future directions”. In: *Computing* 103 (2021), pp. 993–1023.

Payment systems in clouds (models used by Cloud Service Providers) for cloud-based application

Sybil Xie
Department of Computer Science
University of Amsterdam
2757089
c.xie@student.vu.nl

Serein Li
Department of Computer Science
University of Amsterdam
2760463
r.li3@student.vu.nl

Yue Zhang
Department of Computer Science
University of Amsterdam
2772421
y.zhang8@student.vu.nl

Abstract

The way that organizations and people access and use computing resources has been changed by cloud computing. Understanding how payment system models are changing is essential as demand for cloud services increases. In order to provide light on how payment systems have evolved and what effect they have on pricing models and user experiences, this literature review examines both past and present payment system models used in cloud computing. The report explores fixed pricing, dynamic pricing, and value-based pricing, highlighting their advantages, disadvantages, and the importance of cloud service users understanding these models.

1 Introduction

With the introduction of cloud computing, organizations now have access to a wide range of flexible, scalable, and economical services. One crucial aspect of this digital transformation is the implementation of payment systems in clouds, which has become a focal point for Cloud Service Providers (CSPs) [21]. This literature review delves into the various models used by CSPs for cloud-based applications, providing a comprehensive understanding of the current landscape and future trends.

Cloud-based applications cannot work without payment mechanisms, which enable frictionless interactions between consumers and service providers. Over time, these systems have changed to accommodate both the changing needs of businesses and customers. The path has been characterized by technology improvements and creative solutions, from conventional approaches to sophisticated cloud-based systems.

The CSPs' various models for these payment systems each have their own advantages and disadvantages. The "pay-as-you-go" business model is one that some service providers use to give their clients flexibility and scalability. Others prefer a business plan that relies on subscriptions because it provides a more steady and regular flow of income. Also increasing popularity is a hybrid model that incorporates the greatest features of each.

In-depth examination of these models will be done in this paper using case studies from top CSPs including Amazon Web Services, Microsoft Azure, and Google Cloud. The difficulties providers have putting these systems in place will also be covered, along with possible answers.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Cloud-based payment systems have a bright future ahead of them as new trends emerge as technology advances. This assessment tries to clarify these patterns and provide information on how cloud-based applications and associated payment systems will develop in the future.

In conclusion, it is impossible to exaggerate the significance of effective payment mechanisms in clouds. Understanding these systems and their implications is essential for both providers and customers as cloud computing continues to rule the digital environment.

2 Evolution of Payment Systems in Clouds

In the early days of cloud computing, fixed pricing was the predominant pricing model. Customers could choose a plan and then upgrade to a higher tier if it didn't suit their needs. Most cloud services now offer a form of dynamic billing where customers only pay for the resources they have used [10]. Service Level Agreements (SLAs) are a crucial component of cloud computing. They detail the negotiations between the provider and the consumer. Other points included in the SLA are the provision of services, the specific level of Quality of Service (QoS), and guarantees. The final agreements are documented in a contract with the concerned parties [3].

Over time, payment systems began to adopt dynamic models that change based on supply and demand. If a large amount of resources is required at a certain moment in time, the price per unit can rise. The main advantage of this model is that the service provider has more control over the price and can potentially generate more profits. For users, this introduces price risk, as they have to take this into account when they reserve resources to prevent unintentional overpayment. Dynamic systems are also more difficult to implement as they require more features: customers generally want to be able to set upper and lower bounds to reduce their price risk, and algorithms are needed to determine the current market price of the requested resources [15].

Market-dependent pricing models are rarely used and are quite difficult to implement. In this model, the price is not only based on supply and demand like dynamic pricing but a bid-ask marketplace is also introduced, where multiple bidders can buy resources based on real-time market conditions. This strategy can be profitable but is infrequently utilized because it is highly challenging to implement because it requires a working marketplace. We may anticipate more advancements in payment systems with more complex pricing models that better reflect the value offered to customers and the costs incurred by providers as cloud computing continues to expand.

3 Models Used by Cloud Service Providers

3.1 Current models used by cloud service providers

The current models used by cloud service providers Cloud service companies offer a variety of pricing models to suit the various needs of their clientele. Fixed pricing models, dynamic pricing models, and value-based pricing models are the three main categories. Each has unique benefits and potential disadvantages of its own. Customers must therefore be aware of their own requirements and available resources before choosing a model.

Fixed pricing, also known as static or cost-based pricing, is perhaps the simplest and most commonly used model among these. Customers subscribing to this model pay a predetermined fee for a specified service unit. This could be charged monthly, per computing node, per CPU usage, or based on a different metric. The primary advantage of fixed pricing is its ease of implementation.

For businesses seeking transparency and predictability, a clear pricing structure is essential. This can be found in the fixed model, which provides stability around budgeting needs. However, there is a drawback to this option as it doesn't account for fluctuations in supply and demand within the market. The other consideration is dynamic pricing models that base adjustments on supply and demand changes in real time. While this offers flexibility, the user may experience price risk at peak times. On the other hand, service providers can take greater control over their pricing with potential profits that reflect market conditions.

It also necessitates a more complex implementation process, as the service provider needs to constantly monitor and adjust their pricing in line with market conditions [15].

The third model is value-based pricing. It looks at the pricing issue from the perspective of the customer's value perception. Instead of setting a price based on the service cost or the current market conditions, this model derives the price from the perceived value of the service to the customer. This can allow CSPs to maximize their profits by charging a premium for highly valued services. However, implementing a value-based pricing model has its own challenges, for it requires a thorough understanding of each customer's perceived value, which can be highly subjective and difficult to quantify.

3.2 Fixed Pricing Model

Fixed pricing, also known as static pricing or cost-based pricing, is a pricing model where the price is predetermined, providing transparency and stability between the Cloud Service Provider (CSP) and the customer [5]. This model is predominantly used in the early stages of cloud computing, where the market was less volatile and the demand was relatively predictable.

Fixed pricing models are typically adopted by companies that value predictability and stability in their costs. These companies usually have a steady and predictable usage pattern, which aligns well with the fixed pricing model. Major CSPs like Amazon Web Services, Microsoft Azure, and Google Cloud Platform have offered fixed pricing models, especially for their Software as a Service (SaaS) offerings [8].

One of the main advantages of the fixed pricing model is its simplicity and ease of implementation. The price is set in advance, and customers know exactly how much they will be charged, making it easier for them to budget and plan their expenses. This model also provides stability for the CSP, as they can predict their revenue based on fixed prices.

However, the fixed pricing model also has its disadvantages. The main one is that the price might not reflect the actual market value. Customers might end up paying more than the market value if their actual usage is less than the provisioned resources. This model also makes it difficult for providers to change the price, limiting potential profits.

3.2.1 Subscription-based Model

The subscription-based model is a common type of fixed pricing model used in the cloud industry, particularly for SaaS offerings. In this model, customers pay a fixed fee each month or year for a specific level of service. The fee usually depends on the number of users, the amount of resources, or the level of service required.

The subscription-based model is favored by businesses with predictable usage patterns, as it allows for easy budgeting and cost control. It also provides transparency in pricing, as customers know exactly how much they will be charged each month [6].

However, like the general fixed pricing model, the subscription-based model may not be cost-effective for customers with variable or unpredictable usage patterns. Customers may end up paying for more resources than they actually use. Furthermore, this model may not reflect the actual market value of the resources, especially in a rapidly changing market like cloud computing.

3.2.2 Pay-as-you-go Model

The pay-as-you-go model is a payment method that provides users with flexibility by charging them based on their specific usage or consumption of a product or service[2]. This model calculates charges at regular intervals, such as hourly, daily, or monthly, ensuring that users only pay for the resources or services they have actually utilized during that specific period[9]. In 2006, Amazon Web Services recognized the significance of this concept and introduced a pay-as-you-go computing service that required no contractual commitment. Customers simply needed a credit card to access and utilize the services. The key advantage of this model lies in its ability to accommodate changing needs, allowing users to easily scale their usage up or down as required, and adjust their payment accordingly. By aligning costs with usage, the pay-as-you-go model offers a cost-effective and adaptable solution for users.

3.2.3 Tiered and Volume-based pricing

Tiered and volume-based pricing are two distinct yet interrelated pricing strategies that businesses utilize to incentivize larger purchases and optimize revenue.

Tiered pricing is a model where the unit cost of a product or service decreases as the quantity purchased increases. It establishes different price points for different levels or "tiers" of product quantities. For example, a company may charge $100 for 1-10 units, $90 for 11-20 units, and so on. This method encourages customers to purchase in larger quantities to enjoy a lower unit cost. It also allows businesses to cater to a wide range of customers with varying needs and budget constraints.

Volume-based pricing offers a flat discount on the total cost based on the quantity purchased. In this model, the price per unit remains constant, but the total cost decreases as the volume of purchase increases. For example, a company may offer a 5% discount on orders over 100 units and a 10% discount on orders over 200 units. This strategy encourages bulk purchases and can be especially effective in industries where marginal production costs decrease with volume [15].

3.3 Dynamic Pricing Model

The concept of the dynamic model entails that the price of the product or service is determined by multiple dynamic factors, including time, customer demand, and other relevant variables[18]. Many companies employ this pricing strategy to effectively manage their limited capacity or resources. The dynamic model has found extensive application in various service and utility industries. Cloud Computing has transformed a large part of the IT industry, making computing a utility, so developers of innovative internet services no longer face the need for significant upfront capital investment in hardware or extensive human resources for operations. The unprecedented resource elasticity marks a significant milestone in IT history[11].

Talking about payment or pricing models, the elasticity of the dynamic payment model follows the core advantage of Cloud Computing services. There are a variety of dynamic payment models offered by CSPs, providing cloud customers the chance to choose their most satisfying solutions in the marketplace. The dynamic model offers many advantages for both CSPs and customers.

For the CSPs, the flexibility enables them to adapt to market conditions and optimize their pricing strategies accordingly. By adjusting prices, they can incentivize customers to utilize underused resources or shift demand to less busy periods, resulting in better spare cloud capacity utilization and cost optimization. They can also respond quickly to market dynamics, optimize their pricing strategies for more benefits, and stay ahead of competitors, bringing about a competitive edge in the market.

For cloud customers, the flexibility enables a much closer alignment of resources with the workload at hand. Real-world estimates of server utilization in data centers range from 5% to 20%. Another observation is that the peak workloads for many services can surpass the average workload by factors of 2 to 10 [11]. Since few users provision resources for less than the anticipated peak, they end up provisioning for the peak and leaving resources idle during non-peak periods. The greater the variation in workload, the more waste occurs. Elasticity provided by dynamic models can mitigate this waste, compensating for the potentially higher cost per server hour associated with traditional fixed payment models.

For example, AWS offer spot instances or spot blocks with pricing that can be categorized as dynamic-based pricing in 2009. It can be considered as auction-based, cost-based, or time-based pricing due to its multiple characteristics[16]. The shifting conditions of supply and demand have an impact on these pricing alternatives. It might be challenging to classify one pricing model into a single category because the CSPs frequently incorporate components from several pricing strategies into their models. Pricing structures that exhibit traits of many pricing models may be produced as a result of the dynamic nature of cloud computing and the variability in supply and demand.

We will examine two typical dynamic models used for cloud pricing.

3.3.1 Segment-based model

Segment-based pricing is a strategic approach employed by CSPs to categorize their customers into segments based on specific criteria. This strategy allows CSPs to offer customized pricing plans and services that cater to the unique needs and requirements of each customer segment.

The criteria used for segmentation may vary depending on the CSP and its target market. Common factors considered for segmentation include usage patterns, resource demands, geographic location, industry vertical, customer size, and specific service preferences. For example, Microsoft implements segment-based pricing by offering student licenses for their MS Office package, recognizing the specific needs and budget constraints of students. Another example is when a cloud service provider offers discounted prices specifically tailored for students to access cloud services, understanding their unique requirements and financial limitations. Another example is Amazon segments its customers by combining operational revenue streams, such as e-commerce and cloud services, to offer bundled pricing options and provide business customers with valuable advice on cost optimization and efficiency[18].

Segment-based pricing brings several benefits for both CSPs and customers. For CSPs, it allows them to optimize revenue generation by setting different prices based on the perceived value of their services within each segment. This approach maximizes profitability and ensures that customers in high-value segments contribute proportionally to the provider's overall revenue.

From a customer perspective, segment-based pricing offers greater flexibility and cost-effectiveness. Different segments may have varying needs and usage patterns, and pricing plans can be tailored accordingly. This ensures that customers only pay for the resources and services they actually utilize, resulting in a more personalized and cost-efficient solution.

3.3.2 Auction-based model

Auction-based pricing is a dynamic pricing model widely utilized across various industries. Under this model, prices are established through competitive bidding among buyers. Sellers present their products or services, and buyers place bids to acquire them. The highest bidder emerges as the winner of the auction and pays the amount they bid.

Auction-based pricing offers several advantages. The auction process is swift and devoid of intricate processing steps. The pricing mechanism is transparent, with bidders only required to pay the incremental cost at each bid. It also ensures fairness among all participating bidders who adhere to the rules of the auction. However, this model has its limitations. Bidders face time constraints while making decisions during the bidding process, which can result in bids surpassing the actual value of the goods.

In the cloud market, the auction-based pricing model is specifically designed for niche and expanding markets. Its objective is to capture additional value from customers at the lower end of the demand curve. In 2009, AWS introduced its own auction-based spot instance. The AWS bidding process initially functions as a blind auction, where bidders simultaneously submit their prices without knowledge of other participants' offers[18].

The auction-based model in the cloud market allows market forces to determine the price, establishing a transparent and competitive environment. It facilitates price discovery, as buyers' and sellers' willingness to engage in transactions is unveiled through the bidding process. Auctions create a sense of urgency and excitement, fostering competition among buyers and potentially maximizing revenue for sellers.

3.4 Value-based pricing model

The models we previously introduced mainly focus on the cost or the market, but differently, the value-based pricing model is driven by demands.

Value-based pricing is a subjective pricing strategy that focuses on the perceived value delivered to customers, rather than market prices or service costs. It takes into account customers' expectations[13].

Customers' perceived values consist of five dimensions: functional, conditional, social, emotional, and epistemic values. The decision-making process for customers is influenced by these perceived

values. Value-based pricing offers the advantage of capturing a wide range of cloud service values, including emotional and epistemic aspects.[17]

A key challenge is defining value metrics that accurately measure customers' subjective perceptions of value. Hedonic pricing might be a useful model for estimating the value of new services if historical data is available. [19]

4 Case Studies

4.1 Amazon Web Services

AWS is a cloud computing platform that provides a wide range of services to assist individuals, enterprises, and organizations in efficiently and securely maintaining their applications, infrastructure, and data. These services include computing power, storage options, networking capabilities, databases, and machine learning. Furthermore, they cater to numerous industries and use cases with Artificial Intelligence boosting the company's abilities.

Currently, AWS operates on three pricing models basically.

Pay-as-you-go (Fixed Pricing Model) The pay-as-you-go model lets customers adjust resource usage as needed without long-term commitments. Customers using AWS services pay based on actual consumption under this arrangement. Businesses may respond to shifting demand without running the risk of over-provisioning or idling capacity because of this flexibility.

Savings Plans (Fixed Pricing Model) Savings Plans offer cost savings for AWS Compute and AWS Machine Learning services in exchange for a commitment to use a specific amount of service over a one- or three-year period.[4] Customers can get lower prices compared to on-demand rates by committing to a set amount of usage. While there is a commitment associated with Savings Plans, AWS also provides freedom within that commitment. Customers are allowed to use the services they have been given in a flexible way, changing how they use the resources according to their needs.

Volume-Based Discounts (Dynamic Pricing Model) As clients' consumption grows, AWS offers volume-based discounts that result in savings. In the case of certain services such as S3, the price per unit (e.g. per GB) tends to decrease as more individuals avail the service. This kind of pricing system encourages users to consume more resources due to the fact that the price per unit falls alongside increasing volume.

This dynamic pricing model aligns with the concept of dynamically adjusting prices based on usage levels. Customers benefit from economies of scale and pay less as their usage increases, which reflects the dynamic nature of their resource consumption.

4.2 Microsoft Azure's payment system

Azure, Microsoft's cloud computing platform and service, offers a wide range of cloud services that enable businesses to create, manage, and launch apps and services using Microsoft-run data centers. Its offerings include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) tools among several others. With Azure at their disposal, companies can scale the performance of their applications while providing users with great experiences by deploying them closer to their intended audiences. Notably, Azure has an adaptable architecture making it fit for multiple workloads or applications since it supports diverse programming languages, platforms, operating systems, etc. Businesses can select the Azure pricing model that best fits their requirements and financial constraints from a variety of options offered. We list various Azure pricing models, along with their benefits.

Pay-as-you-go (Fixed Pricing Model) This pricing model allows users to pay only for the resources they consume on Azure, without any upfront costs or long-term commitments. It is a flexible and affordable choice for companies with unpredictable workloads or those wishing to test new services because users are charged based on hourly usage rates for the services they use.

Reserved Instances (Fixed Pricing Model) With Reserved Instances, customers can commit to using particular Azure resources for a set amount of time (usually 1 or 3 years) in exchange for a discount off the pay-as-you-go prices. Businesses with predictable workloads and long-term resource needs are suitable to use this model since it allows them to cut expenses while ensuring resource availability.

Hybrid Benefit (Value-based) The primary value of Azure Hybrid Benefit comes from the cost savings it offers to users who already have on-premises Windows Server or SQL Server licenses. By leveraging their existing licenses, users can significantly reduce the costs of running their workloads on Azure virtual machines compared to the standard pay-as-you-go rates[7].

The Hybrid Benefit can be categorized as a cost-saving option in conjunction with other pricing models like Pay-as-you-go. It can also be considered as a value-based pricing option from the perspective of the value it provides to users by maximizing existing investment and offering customized pricing. It delivers value to users, aligning the cost of the service more closely with the users' perceived value.

Spot Instances(Dynamic Pricing Model) This pricing model falls under the category of dynamic models because it enables companies to bid on available Azure capacity, which can save them a lot of money.

4.3 Google Cloud's payment system

The full range of cloud computing services provided by Google is known as Google Cloud. It offers both businesses and people a vast range of information and tools for creating, deploying, and managing their cloud applications and services. Users get access to scalable and trustworthy infrastructure, storage options, machine learning capabilities, data analytics, and more with Google Cloud. Businesses can efficiently innovate, cooperate, and grow their operations thanks to the flexible and secure environment that it provides. Because of its broad ecosystem and wide range of offers, Google Cloud is a well-liked option for businesses, developers, and individuals looking for innovative cloud solutions. The main pricing models available on the Google Cloud Platform are as follows.

Pay-as-you-go (Fixed Pricing Model) In this pricing model, Google Cloud enables users to access and utilize cloud services without any upfront expenses or long-term obligations. This pricing strategy is basically the same as the other two companies we mentioned in previous chapters.

Reserved Instances (Fixed Pricing Model) In exchange for committing to a longer period of service usage, Google Cloud provides users with significant discounts compared to the pay-as-you-go model. Users can commit to using Google Cloud services for a predetermined amount of time by choosing this pricing plan, which enables them to enjoy significant cost savings. Organizations can efficiently save costs while assuring the reliable availability of Google Cloud services by using the longer-term commitment approach. Businesses that can predict their resource needs and are prepared to commit to Google Cloud for a certain period of time stand to gain the most from this strategy.

Free tier plan (Value-based) This plan provides a free tier option for those who are not yet ready to fully transition to a cloud service[12]. It offers a predefined amount of resources over a specific period, allowing users to try out the service without incurring costs. Google also offers "always free" cloud services, which are suitable for organizations with minimal usage requirements and are not concerned about potential interruptions in operations.

By doing this, Google Cloud allows potential customers to experience and evaluate the value of their services firsthand. Google Cloud aims to attract and retain users based on the perceived worth of their offerings, which can be interpreted as having elements of a value-based pricing model.

5 Challenges and Solutions

5.1 Challenges

As cloud computing adoption grows, in order to put in place effective payment systems for its clients, CSPs must overcome a number of obstacles. The main obstacles that CSPs must overcome are covered in this chapter in order to provide their users with a simple and efficient payment experience.

Managing Diverse Pricing Models CSPs face a significant challenge in managing the diverse pricing models available to customers, including pay-as-you-go, subscription-based, reserved instances, and hybrid benefits. Each model has distinct billing and payment requirements, posing difficulties in developing a unified and efficient payment system that accommodates all customer needs

Security Concerns Security and compliance are one of the most paramount concerns for CSPs when it comes to their payment systems. It is crucial for them to maintain the integrity and confidentiality of customer data. To achieve this, CSPs must comply with stringent data protection and privacy regulations, including the General Data Protection Regulation (GDPR) and the Payment Card Industry Data Security Standard (PCI DSS). By adhering to these regulations, CSPs can safeguard customer information, prevent unauthorized access, and mitigate the risks of data breaches.

5.2 Solutions

To address the challenge of managing diverse pricing models, more customer research can be conducted upfront to understand customers' true needs and preferences[20].

In order to solve security issues and ensure the security of the payment system, the payment system can be specially designed to enable regulatory compliance and ensure compliance with relevant data protection and privacy regulations. It is not possible for a company like Amazon Web Services (AWS), for example, to offer its customers a compliance solution. However, what it can and does do is guarantee users that rigorous data-privacy policies are in place and give full disclosure on exactly where a brand's data is stored at all times[1].

6 Future Trends

In the previous section, we discussed the popular models of payment systems and provided some well-known cases of cloud service providers. However, with the continuous advancement of technology and evolving market demands, payment system models are also expected to change. Here are the future trends that we have identified:

Exploring Consumer Self-Selection and Complex Models In the future, there is a potential for consumers to have the ability to self-select from multiple pricing schemes[5]. This has the potential to impact the optimal pricing choices made by service providers. As a result, it becomes crucial to examine the dynamics of complex models involving both providers and consumers. Addressing this evolving landscape would present an exciting and promising area for future research in the field of pricing strategies and models.

Cost-based to Value-based Pricing The traditional approach to cloud pricing has primarily been based on the cost incurred by the cloud service providers. As cloud computing evolves, the importance of considering the value delivered to customers is becoming more apparent. With value-based pricing, customers are able to get the most out of cloud services by taking into account the benefits and outcomes they will receive. This approach involves understanding customers' needs, expectations, and the impact that cloud services have on their businesses. Pricing structures can be established that accurately reflect the worth of providers' offerings by considering the value they create for customers[18].

Emerging Technologies Shaping Cloud Pricing Serverless Computing, Docker Containers, Open API, DevOps, Desktop Grid, Microservices[18], by introducing new technologies, optimizing resource utilization, and enabling more granular and flexible pricing models, these emerging technologies are changing cloud pricing.

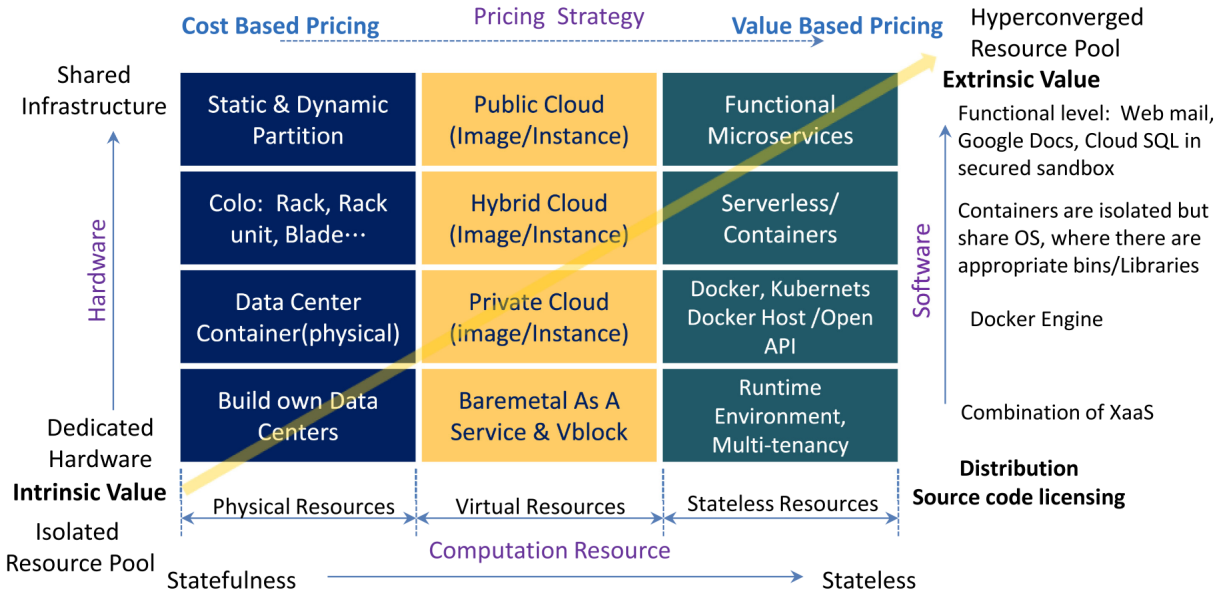


Figure 1: Future trends in cloud technologies and cloud pricing strategy by Wu et al. [18]

Serverless computing involves cloud providers managing the infrastructure and allocating resources automatically without the need for server management. This technology enables developers to focus solely on writing and deploying functions or applications without concerning themselves with the underlying infrastructure. Serverless computing is revolutionizing cloud pricing by introducing a pay-as-you-go model based on actual usage rather than fixed resource allocation.

Docker containers make it simpler to scale and manage applications by enabling uniform application deployment across many environments. The adoption of containerization technologies like Docker is having an impact on cloud pricing models since they allow for effective resource consumption and flexible resource allocation based on application requirements.

Microservices design has an impact on cloud pricing since it enables cost minimization and fine-grained resource allocation by scaling only the essential parts of an application. Complex programs are divided into smaller, deployable services that may be built and scaled separately using a microservices architecture. This method improves the software systems' flexibility, scalability, and fault isolation.

By utilizing these technologies, cloud service providers are better able to meet client requests, allocate resources optimally, and develop creative pricing plans that complement certain service offerings.

7 Conclusion

The terms "cloud computing" and "cloud services" are often used today. We go over the most prevalent cloud payment system models in this review. The evolution of cloud payment systems has changed significantly over time. From the early days of fixed pricing models, where customers paid a predetermined fee for a specified unit of service, to the introduction of dynamic pricing models based on supply and demand adjustments, and even market-dependent pricing models based on real-time market conditions, payment systems have become more complex.

For both users and cloud service providers, fixed pricing models offer transparency and predictability while being straightforward and stable[14]. Overpayments, however, may result if prices do not accurately reflect market values and are not flexible enough to be adjusted.

Dynamic pricing methods give service providers more flexibility over pricing and could result in bigger margins. Customers gain from having the option to increase usage as needed. However, dynamic pricing introduces price risk to users and requires a more complex implementation process.

Value-based pricing models focus on the perceived value delivered to customers and take into account their expectations. By aligning prices with customers' perceived value, service providers can maximize profits. This is the future trend of pricing models for cloud payment systems. Implementing a value-based pricing approach, however, makes it difficult to quantify how people perceive value.

Cloud service providers use a variety of models in these categories, such as subscription-based models, pay-as-you-go models, tiered and volume-based pricing, segmentation-based pricing, and auction-based pricing. Each model caters to various consumer needs and usage patterns and has its own advantages and disadvantages. We may anticipate more advancements in payment systems, with more complex pricing models that better reflect the value offered to customers and the costs spent by providers, as cloud computing continues to expand. The challenge is to find the right balance between simplicity, transparency, flexibility, and profitability to meet the different needs of cloud service providers and customers.

References

[1] The future of payments in the cloud — techradar.com. `https://www.techradar.com/news/the-future-of-payments-in-the-cloud`. Date Accessed: 05-Jun-2023.

[2] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. Deconstructing amazon ec2 spot instance pricing. *ACM Transactions on Economics and Computation (TEAC)*, 1(3):1–20, 2013.

[3] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.

[4] AWSpricing. https://aws.amazon.com/pricing, 2023. Date Accessed: 05-06-2023.

[5] Byong-Sam Chun, Se-Hak; Choi. Service models and pricing schemes for cloud computing. *Cluster Computing 2013-sep 14 vol. 17 iss. 2*, 17, sep 2013.

[6] Se-Hak Chun et al. Cloud services and pricing strategies for sustainable business models: analytical and numerical approaches. *Sustainability*, 12(1):1–1, 2019.

[7] CsharpCorner. https://www.c-sharpcorner.com/article/azure-pricing-models-understanding-the-different-pricing-options/, 2023. Date Accessed: 04-06-2023.

[8] Ilias Daia, Lennart Kerkvliet, Lloyd Nyarko, and Mick Vermeulen. Pricing models in clouds and for cloud-based applications - literature review. Unpublished manuscript, 2023.

[9] Nicola Dimitri. Pricing cloud iaas computing services. *Journal of Cloud Computing 2020-mar 03 vol. 9 iss. 1*, 9, mar 2020.

[10] AWS Economics 2 (EC2). https://aws.amazon.com/ec2/pricing/on-demand/, 2021. Date Accessed: 20-05-2021.

[11] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13):2009, 2009.

[12] GoogleCloud. https://cloud.google.com/free, 2023. Date Accessed: 04-06-2023.

[13] Navendu Jain, Ishai Menache, Joseph Naor, and Jonathan Yaniv. A truthful mechanism for value-based scheduling in cloud computing. *Theory of Computing Systems*, 54:388–406, 2014.

[14] C. N. Höfer; G. Karagiannis. Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications 2011-jun 19 vol. 2 iss. 2*, 2, jun 2011.

[15] Devesh Lowe and Bhavna Galhotra. An overview of pricing models for using cloud services with analysis on pay-per-use model. *International Journal of Engineering & Technology*, 7:248, 2018.

[16] Stamatia Rizou and Ariana Polyviou. Towards value-based resource provisioning in the cloud. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 155–160. IEEE, 2012.

[17] Wolfgang Ulaga and Samir Chacour. Measuring customer-perceived value in business markets: a prerequisite for marketing strategy development and implementation. *Industrial marketing management*, 30(6):525–540, 2001.

[18] Caesar Wu, Rajkumar Buyya, and Kotagiri Ramamohanarao. Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges. *ACM Computing Surveys (CSUR)*, 52(6):1–36, 2019.

[19] Caesar Wu, Adel Nadjaran Toosi, Rajkumar Buyya, and Kotagiri Ramamohanarao. Hedonic pricing of cloud computing services. *IEEE Transactions on Cloud Computing*, 9(1):182–196, 2018.

[20] Amoy X. Yang. Price differentiation model: its challenges and solutions. *Journal of Revenue and Pricing Management 2019-feb 27 vol. 18 iss. 2*, 18, feb 2019.

[21] Fuw-Yi Yang, Chih-Wei Hsu, and Su-Hui Chiu. An e-cash payment system on cloud, 2013.

How does mobile cloud computing contribute to sustainable development and green computing initiatives?

Joshua Offermans
13846183
University of Amsterdam
Amsterdam
The Netherlands
.j.a.offermans@uva.nl

Mattheus Hanna
1473796
University of Amsterdam
Amsterdam
The Netherlands
m.hanna@vu.nl

Maurits Dijk
14743663
University of Amsterdam
Amsterdam
The Netherlands
m.r.dijk@uva.nl

Abstract

Now more than ever do we need to look at finding ways to improve the environmental impact we have on the world. Also in cloud computing do we need to find ways to improve efficiency and decrease the carbon footprint from these technologies. Cloud computing is becoming more ubiquitous everyday and with this growth the demand for energy and other resources is growing as well. There is a clear need for finding less resource intensive ways for computing and mobile cloud computing can as this goal. In this paper we present mobile cloud computing, it's strength and weaknesses, and how it is contributing to the green IT initiative.

1 Introduction

1.1 Background

Mobile cloud computing has emerged as a game-changing model that has transformed the way we use mobile devices and access computational resources. To take around the ambiguity around the term Mobile cloud in this literature study. We define Mobile cloud as dedicated cloud infrastructure for Mobile devices. There is a need for more research on this topic the increasing popularity of smartphones and tablets has led to a growing demand for mobile applications and services. However, these devices face significant limitations in terms of processing power, storage capacity, and battery life, making it challenging to meet these growing demands [12].

To overcome these challenges, mobile cloud computing has created considerable attention as a promising solution. By using the power of cloud computing, mobile cloud computing allows for the offloading of computationally intensive tasks and data storage to remote servers. This improves the capabilities of mobile devices by allowing users to access a variety of services and resources without being limited by the capabilities of their devices. [1].

Meanwhile, the concept of sustainable development and the green computing initiative have become more important in the era of digital transformation. The environmental impact of information and communication technologies (ICT) has raised significant concerns, given their rapid growth and energy consumption. Sustainable development points out the need to create balance between economic growth, social development, and environmental protection. Green computing, a subset of sustainable development, aims to minimize the ecological footprint of ICT through energy efficiency, resource optimization, and responsible technology practices [19].

Taking these considerations in mind, this thesis addresses the research question: "How does mobile cloud computing contribute to sustainable development and the green computing initiative?" This

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

question seeks to explore the potential benefits and implications of integrating mobile cloud computing with the objectives of sustainable development and the practices of green computing. By researching the environmental, social, and economic impacts of mobile cloud computing, this paper aims to shed light on its potential role in promoting sustainability and minimizing the environmental footprint of mobile technologies.

Through a comprehensive analysis of service models, infrastructure requirements, platforms, and applications in mobile cloud computing, this thesis aims to provide valuable insights into the contribution of mobile cloud computing to sustainable development and the green computing initiative. Understanding the potential benefits and challenges associated with this integration can empower policymakers, industry practitioners, and researchers to make informed decisions and develop strategies to leverage the potential of mobile cloud computing in a sustainable and environmentally responsible manner.

1.2 Mobile Cloud Computing: Service Models and Infrastructure Requirements

Mobile cloud computing offers three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS provides virtualized computing resources, like virtual machines, storage, and networking, allowing mobile applications to perform tasks and access infrastructure on-demand. PaaS offers a complete development environment with built-in services for mobile app development, while SaaS delivers fully functional software applications to mobile devices over the cloud.

To support mobile cloud computing, infrastructure requirements include cloud data centres, wireless networks, and edge computing. Cloud data centers host servers, storage, and networking equipment, providing resources for data processing, storage, and service delivery. Wireless networks, such as cellular and Wi-Fi, connect mobile devices to the cloud infrastructure, enabling access to cloud services and data exchange. Edge computing brings computation and storage closer to mobile devices, reducing latency and improving responsiveness by processing data near the source.

In short, mobile cloud computing's service models (IaaS, PaaS, and SaaS) enable the delivery of services to mobile devices, while infrastructure components like cloud data centers, wireless networks, and edge computing support the computational and storage needs of mobile applications.

1.3 Platforms and Applications Analysis

The analysis of platforms and applications in mobile cloud computing is essential for understanding their implementation and usage. Mobile cloud computing platforms enable the development, deployment, and management of mobile applications in the cloud. They provide developers with tools and resources to optimize mobile apps using cloud services.

Platform analysis considers factors like ease of use, scalability, security, and compatibility with different mobile operating systems. It also focuses on applications across domains like healthcare, e-commerce, entertainment, and productivity. Understanding their use cases and requirements provides insights into the advantages and challenges of integrating mobile cloud computing into diverse industries.

Performance and user experience are crucial. Factors like response time, data transfer rates, and reliability impact the success and usability of mobile cloud applications. Real world case studies and user feedback offer valuable insights into their performance and satisfaction levels.

1.4 Mobile Cloud Computing and Sustainable Development

Sustainable development is all about finding a balance between economic growth, social progress, and protecting the environment. In the context of mobile cloud computing, it's important to understand how it contributes to sustainability and its impact on society and the environment.

This paper will focus on the environmental side of mobile cloud computing and how it can help reduce the environmental impact of mobile technologies. It explores how offloading computational tasks to remote servers can save energy, which means our mobile devices will use less power and last longer on a single charge. Also it will take a look into how cloud data centers can adopt green practices like using energy-efficient hardware and renewable energy sources.

It will also dive into the social and economic effects of mobile cloud computing on sustainable development. investigating how it can make advanced services more accessible and affordable for everyone, promoting inclusivity and also mentions the economic benefits for businesses and individuals, like saving money, boosting productivity, and creating job opportunities.

2 Research

2.1 Research questions

To answer our main research question How does mobile cloud computing contribute to sustainable development and green computing initiatives? We will divide our research into the following sub-questions: How does Mobile cloud work and how does it distinguish itself from other cloud solutions? In what kind of applications is Mobile cloud used? What is the environmental impact of traditional cloud and how does mobile cloud compare? What are the barriers and challenges of implementing Mobile cloud?

2.2 How does Mobile cloud work and how does it distinguish itself from other cloud solutions?

As mentioned in the introduction mobile devices face shortcomings in the aspect of processing power, storage capacity, and battery life. The goal of mobile cloud is to mitigate these shortcomings and it does so in the following ways. Lowering the latency, minimising data transfer and faster response times. This is achieved by applying the following strategies. Network bandwidth strategy: using regional data centres closer the mobile broadband. Network latency strategy: Move the nodes that need to process on request closer to the mobile broadband. Mobile cloud application elasticity: Dynamically optimizing the application for delivery and execution between device and network [19]. On the topic of application elasticity offloading plays a vital role in mitigating the problem of the lesser amount of computational resources available on a mobile device. Heavy computational issues like speech recognition, natural language processing, computer vision and graphics, machine learning, and augmented reality will then be offloaded to the mobile cloud infrastructure [5].

2.2.1 Alternative cloud programming models

As mentioned earlier, data efficiency is crucial for mobile devices due to bandwidth constraints. In order to improve data efficiency, alternative programming models can be explored, as the currently dominant method of communication on the web is REST (Representational State Transfer). According to a recent survey conducted by Postman, 89% of developers use REST as their API architecture style (multiple answers were possible) [25].

However, REST is known to retrieve more data than necessary and introduce unnecessary overhead. This has led to the exploration of other protocols, such as MQTT (Message Queuing Telemetry Transport), for the development of mobile cloud solutions. In a study that compared MQTT to HTTP in combination with REST, MQTT was found to be 300% faster than its counterpart [3].

By adopting MQTT or similar protocols, mobile cloud solutions can achieve higher data efficiency and reduce unnecessary overhead, leading to improved performance and faster data transfer between mobile devices and the cloud infrastructure.

2.3 Mobile cloud applications

To get a better understanding of how Mobile Cloud distinguish itself from traditional Cloud Computing. There can be looked at what kind of applications Mobile Cloud is used. A survey conducted by researchers at Nanyang Technological University supplies a list of applications where the Mobile cloud is used[10]. A few of those applications are listed below and reviewed more in-depth also other applications are added to the list gathered from other sources.

2.3.1 Mobile Cloud Computing and IoT

The internet of things consists of three main parts: the devices collecting the data, the communication networks that connect everything and the system using the data. IoT suffers from very similar

problems as regular mobile devices. Mainly limited resources. The sheer volume of data is pushing the boundaries of communication and network technology [6]. The mobile cloud allows IoT devices to outsource data processing and computation through the internet allowing for more efficient use of the IoT device and basically unlimited storage. In addition, since cloud computing can be implemented scalable, users can pay based on their demand and the computation can take place in a different geographical location providing access to people in less developed areas of the world [23].

Green IoT Green IoT is at the intersection of Green IT and the Internet of Things. Green IoT has two goals. The first one is to minimize the negative environmental impact of ICT by trying to develop new technologies that can save information and communication energy. The second goal is to use this technology to solve environmental issues.By having the cloud act as a front end for IoT, it is very similar to the way mobile cloud operates [26]. Cloud computing represents two major trends in information technology, one of which is IT efficiency where power is utilized more efficiently through scalable hardware and software resources [23]. Mobile cloud computing allows IoT to take advantage of what cloud computing has to offer. Mobile cloud offers flexible and scalable service, unlimited data storage, improved performance with data processing and extended battery performance.

Edge computing Edge computing is an extension of cloud computing that deploys computing and storage resources at the edge of the network closer to the mobile devices. It does this by utilizing edge devices without uploading to the central cloud platform. This results in faster transferring of data due to it being closer to the data source [8]. This makes it particularly useful for mobile cloud computing. The lower latency allows for faster offloading and thus a larger portion of the data can be offloaded, which can help save energy and reduce overall latency of the system [22].

2.3.2 M-commerce

An application where Mobile cloud plays an important role especially if looked at in terms of revenue is E-commerce or as it is also called when used from a mobile phone M-commerce. Especially convenience attracts users to mobile shopping as shown in research [32] 76% give this as their main reason. If looked at it in terms of revenue the data shows that M-commerce is becoming a bigger and bigger part of total retail sales. In 2022 M-commerce was 6.0% of the total retail revenue and growing with an expected share of 6.5% in 2023 [17]. This shows especially how dominant mobile applications are becoming in the retail space. With these mobile applications also the mobile cloud infrastructure behind them become more important for the vendors because it makes a growing portion of their revenue.

2.4 Healthcare

Another example where mobile cloud computing can play a significant role is in healthcare applications. With the increasing prevalence of mobile devices, numerous opportunities arise, particularly in the healthcare sector, especially when mobile devices are connected to specialized sensors. In research conducted on this topic, positive results have been observed in both patients and healthcare practitioners [7].

The researchers specifically reviewed promising applications in the areas of vital sign monitoring and blood glucose monitoring. In these applications, mobile cloud infrastructure could play a vital role, particularly in real-time AI analysis and providing feedback on the collected data. By offloading computationally heavy tasks to the mobile cloud infrastructure, the processing capabilities of the mobile device can be augmented.

However, the research also emphasizes the need for caution when it comes to the use of mobile applications in healthcare. Given the critical nature of healthcare, it is essential to prioritize reliability, efficiency, security, and privacy when developing such applications, considering the sensitive nature of the data being handled.

2.4.1 Mobile gaming

Mobile gaming is a rapidly growing industry that has emerged as a dominant form of gaming. Research in this field highlights the increasing popularity of mobile gaming and also explores the

challenges and disruptions it faces. One significant factor contributing to the rise of mobile gaming is the improvement in internet speed and the advancement of online infrastructure [11].

Considering this information, it can be assumed that in order to further expand user bases in the future, it is crucial to have adequate cloud infrastructure capable of accommodating even more high-performance games than those currently available on the market. This emphasizes the need for robust and scalable cloud systems that can handle the increasing demand for mobile gaming.

In particular, cloud infrastructure for mobile gaming should prioritize low latency. Low latency is of utmost importance in online gaming, as it directly impacts the responsiveness and real-time interaction between players. To ensure a seamless and enjoyable gaming experience, the cloud infrastructure should minimize network delays and provide efficient data transmission between players and game servers.

By investing in cloud infrastructure with low-latency capabilities, the mobile gaming industry can enhance multiplayer experiences, reduce lag, and provide a competitive advantage to game developers. Furthermore, an efficient and reliable cloud infrastructure can also facilitate cross-platform gaming, enabling players to seamlessly switch between mobile devices, consoles, and PCs without losing progress or experiencing significant disruptions.

In conclusion, the growing popularity of mobile gaming calls for the development of robust cloud infrastructure that can support the increasing demands of high-performance games. Emphasizing low latency in the cloud infrastructure will be crucial in providing a seamless and immersive gaming experience for mobile gamers.

2.5 What is the environmental impact of traditional cloud and how does mobile cloud compare?

The environmental impact of traditional cloud computing and its comparison to mobile cloud computing are crucial factors in evaluating the sustainability of these technologies. Traditional cloud computing relies heavily on large scale data centers, which present significant challenges in terms of energy consumption, carbon emissions, and resource use. In contrast, mobile cloud computing has the potential to reduce the environmental impact by leveraging centralized resources and offloading computational tasks from individual mobile devices.

2.5.1 Environmental Impact of Traditional Cloud Computing

The structure of traditional cloud computing, which is defined by data centers that include many servers, requires a significant amount of electricity to operate and cool, leaving a huge carbon footprint. The ICT sector, which includes data centers and telecommunication networks, is responsible for around 2% of the world's greenhouse gas emissions, which is equal to the emissions of the aviation sector, according to a research by Greenpeace [15]. Concerns about energy use and carbon emissions have been increased by the expansion of data centers and the rising demand for cloud services.

For cooling purposes, data centers also need a lot of water. The significant water footprint of data centers can worsen the world's rising water shortage problem. To reduce the environmental impact of data center operations, effective water management technologies and methods are necessary. Implementing advanced cooling techniques, such as liquid cooling and reclaimed water systems, can significantly reduce water consumption in data centers.

As said before, large-scale data centers are the core for traditional cloud computing, which uses a lot of power. Numerous servers are stored in these data centers, which need constant electricity. The energy demand of data centers contributes to greenhouse gas emissions and strains global energy resources. According to a report by the International Energy Agency (IEA), data centers consumed around 205 terawatt-hours (TWh) of electricity in 2020, accounting for about 1% of global electricity consumption [18]. This energy consumption is projected to continue growing with the increasing demand for cloud services.

2.5.2 The potential of mobile cloud computing

On the other hand, mobile cloud computing has the potential to help the environment by lowering the energy usage of individual mobile devices. The processing, storage, and battery life of mobile

devices like smartphones and tablets are constrained. Mobile cloud computing makes use of remote cloud servers to offload expensive operations, resulting in energy savings and longer battery life for mobile devices [31]. The entire energy use and carbon emissions connected with mobile computing could be reduced by this offloading procedure

Mobile cloud computing can also help reduce electronic waste by extending the lifespan of mobile devices. Traditional cloud computing requires powerful local devices to handle complex computations. However, this can lead to faster device obsolescence and increased electronic waste. By offloading computational tasks to the cloud, the processing capabilities of mobile devices are not strained, leading to lower wear and tear on device components. This can result in longer device lifetimes, reducing the need for frequent device upgrades and replacements.

Additionally, by combining resources in data centers, mobile cloud computing could use green computing techniques. Green computing aims to optimize energy efficiency and minimize the environmental impact of IT infrastructure. By using energy-efficient hardware, such as servers equipped with low-power processors and advanced power management features, data centers can effectively reduce energy consumption [30]. Server virtualization is another method which could enhance resource usage and energy efficiency. Studies have demonstrated that virtualization technology in data centers could reduce energy consumption up to 30% [21].

2.6 Security Challenges

The adoption of renewable energy sources plays a crucial role in mitigating the environmental impact of cloud computing. Leading cloud providers have made commitments to powering their data centers with renewable energy. For instance, Google has successfully matched 100% of its worldwide operations with renewable energy, including its data centers [14]. Similarly, Amazon Web Services (AWS) has made substantial investments in renewable energy projects to power its infrastructure [4]. These initiatives significantly contribute to the overall reduction of carbon emissions associated with cloud computing.

To address the environmental impact of cloud computing, various projects and guidelines have been established. The Green Grid, an industry consortium, focuses on promoting energy efficiency and sustainability in data centers [16]. The European Code of Conduct for Data Centers provides guidelines and best practices for improving energy efficiency and reducing environmental impact in data center operations [20]. These programs seek to encourage the use of sustainable practices and raise awareness regarding the environmental implications of cloud computing.

In conclusion, traditional cloud computing has environmental issues due to its energy consumption, carbon emissions, and resource usage. However, mobile cloud computing offers potential benefits in terms of reducing environmental impact. By leveraging centralized resources, offloading computational tasks, implementing green computing practices, and utilizing renewable energy sources, mobile cloud computing can help to save energy and could decrease carbon emissions. Further research is required to improve energy efficiency, increase the share of renewable energy in data centers, and ensure the widespread adoption of sustainable practices throughout the mobile cloud computing ecosystem.

2.7 What are the challenges and barriers of implementing a mobile cloud?

Mobile cloud computing faces a number of challenges. The major challenges that mobile cloud computing face are inherent problems from mobile devices and the mobile nature of their application in daily life. These range from limited processing resources due to the small form factor to availability issues due to network fluctuations. There is still a lot of research to be done improving on the shortcomings of mobile cloud computing. If we can overcome, or at least diminish, some of these aspects this could help increase adoption of mobile cloud computing and thereby help support sustainable IT.

2.7.1 Energy efficiency and limited bandwidth

Even though mobile cloud computing can improve mobile devices it still has a problem with limited resources available [27]. Energy efficiency is a very important topic when looking at mobile cloud computing. Mobile devices have limited electricity available to perform tasks and it is therefore a

key resource that needs to be considered carefully. One of the main things to consider when dealing with the limited capacity is deciding whether the benefits outweigh the costs of offloading. There are conflicting objectives at play with mobile devices [13]. These three goals are performance, battery lifetime and quality of the data. While having high performance and fast execution is nice to have, this also compromises energy usage and therefore battery life. And many low-resource mobile devices might only be able to provide lower quality data transfer due to severely lacking bandwidth. There is a fine balance to be found in between these three goals to maximize the experience. Where with some applications slightly more performance would be preferable at the cost of decreased battery life. Similarly, because we cannot just connect an internet cable to a mobile device. This means that mobile devices have limited bandwidth available, compared to traditional cloud solutions, to connect with cloud services. These limitations in resources form the biggest challenges but mobility causes other problems. These are availability [10] and heterogeneity [24].

2.7.2 Service availability and Heterogeneity

Mobile cloud services need to be able to support this mobility while providing seamless service to its users [12]. With traditional cloud services the users access it from a known location where the conditions can be optimized for the user. Mobile devices need to be able to provide service from basically anywhere which increases the complexity of the system. Mobile devices could face changes in environment at any given moment [28]. Connection issues are ever present in a mobile environment. While moving a mobile device could change networks many times in short period, for example while moving in a train or car. The connection could go from strong to weak, and vice versa. Or it could disconnect altogether. While going from a strong to a weak connection might be inconvenient it does not pose any serious threat. Tasks could still be executed, although slower. However when a user moves effectively out of range of the available service network during the execution of a tasks, and a complete disconnect happens, the tasks is aborted. This is not only inconvenient for the user, but it also wastes resources. These environmental changes are problematic for the simple reason that we do not, and cannot, know every network. This heterogeneity can make it difficult to maintain an adequate level of service for the users. In addition mobile clouds need to be able to provide service to many different devices, networks and data formats [2].

2.7.3 Security and privacy

Finally mobile clouds face additional security challenges on top of the ones that come with traditional cloud computing. These challenges are: privileged user access, regulatory compliance, physical location of the data, data separation between different users, proper recovery mechanisms and long-term availability [12]. The point of mobile cloud services is to move resources intensive tasks away from the mobile device. As a result of this, data needs to be moved to and from the cloud provider for storage and processing. This process introduces a risk of data loss, data breach, data recovery, data locality and data privacy [24]. Due to the dynamic nature of mobile devices mobile clouds can be accessed from many different unknown and wireless networks which increase the security risks [12, 29]. This can be problematic since the users often are transferring data from a personal mobile device. Data that would otherwise have remained private. Users have no control over the offloading process. This means there is an increased risk of unauthorized access to the data, a data breach. Which increases the potential of violating the integrity, confidentiality and availability of the data [2]. There have been offloading algorithms proposed that would try to mitigate this risk by only offloading less sensitive data [9]. Furthermore, because the users have no control over this process they are unaware of any potential data that could be collected while offloading. The cloud providers could be scanning the data that is being offloaded or they could be recording location data. The sensors on mobile devices allows the cloud providers to give the users context, like the current location of the user, but it also raises additional privacy concerns over the collection and processing of this data [24].

3 Discussion

In this paper, we explored the field of mobile cloud computing across various applications, along with potential environmental benefits. Our research delved into several resources, yet we encountered a challenge: a noticeable lack of recent research on mobile cloud computing. Most existing literature dates back to around 2012.

The likely reason behind this research gap is the significant advancements in mobile performance and bandwidth over the years, which have mitigated many previous problems and diminished the immediate need for new solutions through research.

However, certain areas still warrant further investigation. For example, the role of AI in mobile applications, which demands high computational power and resources, would benefit from more focused research. Likewise, the Internet of Things (IoT) is another area ripe for more extensive study, given the limited resources available for such devices.

Lastly, the issue of energy efficiency in mobile cloud computing requires further attention, particularly in light of the ongoing climate crisis. Future research should focus on how mobile cloud computing can be leveraged to further enhance energy efficiency and contribute to global sustainability efforts.

4 Conclusion

The mobile cloud is a powerful technology that can significantly enhance the capabilities of mobile devices. Mobile cloud computing can extend battery lifetime off mobile devices by offloading complicated processes and computations. Mobile cloud allows us to leverage increased processing power and storage capacity without increasing the cost which can improve reliability and scalability. And by using renewable energy mobile cloud computing can help save energy and decrease the emission of greenhouse gases. It involves unique strategies and models to overcome the limitations of mobile devices, but it also faces its own set of challenges. These challenges involve the limited resources associated with mobile devices and the dynamic nature of mobile devices. Nevertheless, with ongoing advancements in technology and security practices, the mobile cloud will continue to play a critical role in the evolution of mobile computing and green IT.

References

[1] Khadija Akherfi, Michael Gerndt, and Hamid Harroud. Mobile cloud computing for computation offloading: Issues and challenges. pages 1–16, 2018.

[2] Samaher Al-Janabi, Ibrahim Al-Shourbaji, Mohammad Shojafar, and Mohammed Abdelhag. Mobile cloud computing: Challenges and future research directions. *2017 10th International Conference on Developments in eSystems Engineering (DeSE)*, 2017.

[3] Harry Kasuma Aliwarga, Alwy Herfian Satriatama, and Bruno Fandi Adi Pratama. Performance comparison of fleet management system using iot node device based on mqtt and http protocol. In *AIP Conference Proceedings*, volume 2217, page 020009, 2020. Published Online: 14 April 2020.

[4] Amazon. Amazon investing in 274 renewable energy projects globally, adds 18 new projects in europe and u.s., 2021.

[5] Paramvir Bahl, Richard Y Han, Li Erran Li, and Mahadev Satyanarayanan. Advancing the state of mobile cloud computing. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 21–28, 2012.

[6] Mehdi Bahrami, Arshia Khan, and Mukesh Singhal. An energy efficient data privacy scheme for iot devices in mobile cloud computing. *2016 IEEE International Conference on Mobile Services (MS)*, 2016.

[7] Mirza Mansoor Baig, Hamid GholamHosseini, and Martin J. Connolly. Mobile healthcare applications: system design review, critical issues and challenges. *Australasian Physical & Engineering Sciences in Medicine*, 2015.

[8] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, May 2020.

[9] N. M. Dhanya and G. Kousalya. Adaptive and secure application partitioning for offloading in mobile cloud computing. *Communications in Computer and Information Science*, page 45–53, 2015.

[10] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, 13(18):1587–1611, 2011.

[11] Claudio Feijoo, José-Luis Gómez-Barroso, Juan-Miguel Aguado, and Sergio Ramos. Mobile gaming: Industry challenges and policy implications. *Telecommunications Policy*, 2012.

[12] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.

[13] J. Flinn, S. Park, and M. Satyanarayanan. Balancing performance, energy, and quality in pervasive computing. *Proceedings 22nd International Conference on Distributed Computing Systems*, 2002.

[14] Google. Google - clean energy, 2023.

[15] Greenpeace. Clicking clean: Who is winning the race to build a green internet? `https://www.greenpeace.de/publikationen/20170110_greenpeace_clicking_clean.pdf`, 2017. [Online; accessed May 24, 2023].

[16] The Green Grid. The green grid.

[17] Insider Intelligence. Mobile commerce 2023: Latest trends and statistics, 2023. [Online; accessed 5-June-2023].

[18] International Energy Agency. Data centres and data transmission networks. `https://www.iea.org/reports/data-centres-and-data-transmission-networks`, 2022.

[19] Gaurav Jindal and Manisha Gupta. Green computing "future of computers". *Journal of Emerging Research in Management and Technology*, 2012.

[20] European Commission Joint Research Centre. Code of conduct on energy efficiency for data centres.

[21] N. Satish Kumar and B. Karunakar Reddy. Implementation of server virtualization to build energy-efficient data centers. *International Journal of Computer Applications*, 94(8):6–11, 2014.

[22] Mohammed Maray and Junaid Shuja. Computation offloading in mobile cloud computing and mobile edge computing: Survey, taxonomy, and open issues. *Mobile Information Systems*, 2022:1–17, 2022.

[23] S Marston, Zhi Li, S Bandyopadhyay, and A Ghalsasi. Cloud computing - the business perspective. *2011 44th Hawaii International Conference on System Sciences*, 2011.

[24] Muhammad Baqer Mollah, Md. Abul Azad, and Athanasios Vasilakos. Security and privacy challenges in mobile cloud computing: Survey and way ahead. *Journal of Network and Computer Applications*, 84:38–54, 2017.

[25] Postman. State of the api: Api technologies, 2023. Accessed: 2023-06-03.

[26] K.E. Psannis, S. Xinogalos, and A. Sifaleras. Convergence of internet of things and mobile cloud computing. *Systems Science and Control Engineering*, 2(1):476–483, 2014.

[27] Han Qi and Abdullah Gani. Research on mobile cloud computing: Review, trend and perspectives. *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, 2012.

[28] MingJian Tang and Jinli Cao. A dynamic mechanism for handling mobile computing environmental changes. *Proceedings of the 1st international conference on Scalable information systems; - InfoScale '06*, May 2006.

[29] Yating Wang, Ing-Ray Chen, and Ding-Chau Wang. A survey of mobile cloud computing applications: Perspectives and challenges. *Wireless Personal Communications*, 80(4):1607–1623, 2014.

[30] Yu Wang, Bingpeng Zhou, Yang Xiang, Wei Xu, and Yuanyuan Zhang. Energy-efficient server virtualization for mobile cloud computing. *Cluster Computing*, 25(1):681–689, 2022.

[31] Jianlong Xu, Guiyi Wei, Xiaowen Chu, and Zili Shao. Energy saving in mobile cloud computing. 2014.

[32] Dynamic Yield. State of personalization in mobile commerce, 2023.

Critical Factors to the Adoption of Identity as a Service (IDaaS) in Organisations: a Literature Review

Mauricio Bernardo da Silva
Faculty of Science
University Van Amsterdam
Amsterdam, Netherlands

Elif İpek Uysal
Faculty of Science
University Van Amsterdam
Amsterdam, Netherlands

Daniel Yazbek
Faculty of Science
University Van Amsterdam
Amsterdam, Netherlands

Abstract

Identity as a Service (IDaaS) is a model for identity management aligned with the paradigm of cloud computing: as a third-party standalone service offered to enterprises, readily available, flexible, convenient and elastic. From its inception as proposed architectures for identity management, to a growing industry in itself, it has gone through many definitions and iterations. In this literature review, we offer a historical and contextual analysis of IDaaS from 2007 to 2023, focusing on the factors that hinder or enable its adoption in organisations. Evidence is given that security and trust, complexity and lack of compatibility are widely mentioned hindrances, as breaches and outings on sensitive processes such as authentication and authorization are major risks to organisations. On the other hand, the perceived relative advantage and reduction in costs, including speed and ease of set-up and management (being offered by a third party) are incredibly valuable. The portability and interoperability of these systems are very beneficial for organisations and users alike, and if these systems are correctly set up, security and trust can be massively increased. Finally, this work demonstrates that literature specific to this topic is lacking, and as a rapidly growing and mature service model, more attention should be given to the subject.

1 Introduction

Identity as a Service (IDaaS) is a term closely associated with cloud computing. In cloud computing, there are three comprehensive service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IDaaS is a subset of the Software as a Service category. [1] IDaaS was originally proposed as an identity management architecture that would enable the interoperability of many different services in a service-oriented architecture, following the paradigm of loose coupling of identity management with the core services. [2]

The advantages provided by utilising IDaaS are linked to the main selling points of cloud computing: "flexibility, scalability of resources, reliability, broad network access, cost-effectiveness and sustainability". [3] Specifically to IDaaS, the added promised advantages are the convenience for the end-user, related to a single sign-on approach or on every access; as well as interoperability for easy integration of business functionality on the side of organisations. [2]

Although cloud computing rapidly grew from its early development from Amazon in 2007 (IaaS), Salesforce (PaaS) and Google in 2010 (SaaS), until recently, IDaaS was a nascent yet promising portion. [1]. Therefore, the factors that impact the adoption of this specific model of Identity Access Management (IAM) in organisations have not been studied at length.

Overall, five key capabilities are required to make enterprise IDaaS solutions possible [3]:

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

- Single Sign-on (SSO)
- Multi-factor Authentication (MFA)
- Access Security
- Directory
- Provisioning

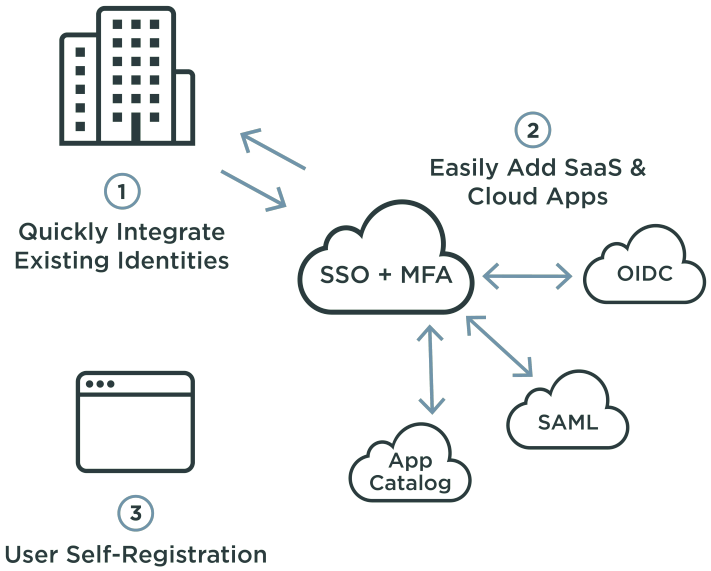


Figure 1: Identity as a Service (IDaaS) [15]

The goal of this literature review is to identify and analyse the relevant scientific literature when it comes to the decision-making process that leads organisations to adopt or not adopt IDaaS as the IAM model for their case. This paper will also include, where relevant, a review of factors in the adoption of cloud computing at large, as IDaaS is a subset of this service model.

2 Research method

We utilised Google Scholar as our primary search tool. Our search included the keywords “Identity as a service”, “IDaaS”, “Cloud Computing” and “Adoption”, either standalone or in combination with one another. We also browsed the following databases:

- IEEE Xplore
- San Jose State University, Dr. Martin Luther King, Jr. Library

The selection criteria were to focus on papers that provided historical context to the development of IDaaS, Cloud Computing and Cloud Services; papers assessing opportunities and challenges of the aforementioned topics; and papers analysing factors for their adoption in organisations. The review will follow a chronological order.

3 Literature review

The term Identity as a Service (IDaaS) was first proposed by Emig et al. in 2007 in their work “Identity as a Service - Towards a Service-Oriented Identity Management Architecture”. They predicted that service-oriented architectures (SOA) would be the basis of future information systems,

and highlighted the lack of a blueprint to create an identity management (IdM) architecture tailored to SOA. [2] With a major advantage of web-service-oriented architecture being the interoperability of services, the IdM need to account for that, facilitating authentication and authorization of users across services.

Their main contribution is to define that "the complexity of the IdM architecture is encapsulated at a set of service interfaces which should not have business domain-specific characteristics.", aligning the IdM architecture with the paradigm of SOA, and then proposing and testing an initial design to validate it. Hence, the IdM is designed as a decoupled service: IDaaS. However, as a starting proposition for IDaaS, their work did not include or predicted any of the commercial ramifications, such as companies offering IDaaS as a third-party alternative, making IDaaS a de-facto product in the cloud ecosystem. With the growth of digital data generation and storage, and the growing need for trust and security, some authors propose an essential shift in how digital identities are maintained and utilised on the internet.

Ates et al. proposed an identity-centric internet in 2011, as a way to defragment personal data. [4] Not just as a matter of decentralised storage, but of multiple systems containing just fragments of a person's information, often without that person's knowledge or consent.

Although this work is tangential to our research question, it provides historical context and describes some challenges in identity management, from societal and technical perspectives. Mainly, the authors bring attention to the notion of ownership of personal data, a topic closely linked with authentication and authorisation, since forms of authentication often require personal information, and that data generated while a user is signed in is often linked to the individual, even if they don't own it.

An interesting proposition in this paper is the Identity in the Cloud Agent (IC-Agent), which is a user-owned and controlled logical system to manage an entity (e.g. person) identity, including personal data, on the cloud.

As interesting as that idea might be, and the claim by the authors that it is a "necessary and tangible objective", it did not come close to materialising since. The security and technical issues when it comes to hosting and integrating such a solution were not addressed at length (except for a brief discussion on the hosting for the IC-Agent). Furthermore, the authors did not address users' motivation in technology adoption, nor the very diverse cultural and regulatory landscape across the globe when it comes to personal data management.

In Samlinson & Usha (2013), it is demonstrated that "cloud computing is the next wave of information technology revolution", but that security is a major hurdle in its adoption.[5] To adapt to the cloud model, organisations would be forced to move away from traditional security methods in-premises and move to higher-level security. Expanding on the idea of Ates et al., they proposed a Trust Agent to allow users to log into multiple cloud service providers (CSP) - a federated environment. Briefly described, the IdM would allow users to "securely traverse in the federated cloud environment without creating new identity credentials for each service they access".

The value add of this work is the description and diagraming of how these Trust Agents can enable identity management in an intercloud scenario, and that it can be achieved with open standard protocols. Additionally, it demonstrates the need for convenience and trustworthiness in this scenario. However, this proposed model was not empirically verified, as the authors suggest it be done in future work.

Stieninger et al. (2014) offer a more comprehensive review of drivers and barriers to the adoption of cloud services. They utilise and compare theories for innovation and IT adoption such as Diffusion of Innovations (DoI) theory by Everett Rogers (1962), from his book of the same name; and the Technology Acceptance Model (TAM) by Fred Davis (1987). Importantly, they acknowledge that, although Davis' TAM focuses on individuals instead of organisations, many IT decisions in organisations are made by single individuals at the executive level. Therefore, it's acceptable to use both these theories to investigate organisational disposition to these factors.

Their paper focuses on and reconceptualises five factors accepted as influential in technology adoption: compatibility (CPT), relative advantage (RA), complexity (CPX), image (I) and security & trust (ST).[6]

- Compatibility: the degree to which the innovation is perceived as consistent with the "existing values, past experiences and needs of potential adopters." It can also encompass process and data compatibility, and considerations of lock-in effects on a vendor on costs for migration and integration. According to the authors, this factor is frequently cited.
- Relative advantage: "how much is the innovation perceived as being better than the idea it supersedes". In this, cloud computing solutions showcase several relative advantages of simpler administration, potential cost savings and flexibility, among others.
- Complexity: the perceived difficulty to understand and use the innovation. High complexity is likely to turn into a barrier to adoption, but there were indications that subject matter experts consider cloud implementation as not very complex, as it has simple administration tools, and high usability and degree of automation.
- Image: the perceived enhancement of the organisation's image or status in its social context. A good image can offer a competitive advantage in attracting and negotiating with customers. Because of association, an organisation can be at risk of negative impact due to issues with the cloud service provider, such as data loss incidents, outages, or even sharing cloud resources with a malicious organisation.
- Security & trust: for their work, trust is "considered as the ability of the involved actors to convey the perception of trustfulness". Due to its novelty at the time, there was still a lack of trust, and perceived security and safety heavily influence trust in the context of cloud services. In particular, we highlight data security and geographical location for data storage and processing as impactful items for trust.

The aggregation of many factors presented in previous theories, and its reconceptualisation, is particularly useful here. The authors compare a somewhat extensive body of research and cite studies that map these factors to positive and negative attitudes toward the adoption of cloud computing. In relation to our original question of factors influencing the adoption of IDaaS, it goes beyond the previously mentioned issues of convenience and security, demonstrating that the organisational context and perception of the innovation play an important role. In this paper, the approach to organising and counteracting some gaps in previous theoretical models is logically explained. However, the findings are related to the broader topic of cloud computing, and there are no specifics on IDaaS, so they have to be interpreted in a more general way.

In a more narrow scope, Habiba et al. (2014) explore cloud identity management security issues and solutions. They define cloud Identity as a Service (IDaaS) as "the management of identities in the cloud, outside the organisational boundary and applications that use them." [7]

They point to the benefits that IDaaS offer that are common to cloud services, such as reduced hardware cost and easier integration, but many challenges exist in security and privacy. They advocate that identity management is still best to be managed internally because the risks lie on the organisation, and not the IDaaS provider, if critical information is lost or compromised.

In the paper, many models of identity management systems are classified and explained, with their respective advantages and disadvantages, as well as a list of open-source cloud computing platforms alongside their identity management services.

As a factor inhibiting adoption, the security of Cloud IdMs was at the time in its nascent stages, and it was "categorically considered as the most important requirement". Security and performance bottlenecks limited adoption in a dynamic cloud environment. The authors bring attention that the domain of cloud IdMs warranted more attention from the research community and IT industry.

The added value of this research was comprehensive research on cloud IdMs, with a classification of models, an in-depth list of possible attacks targeting cloud IdMs, and the development of a taxonomy to evaluate cloud-based IdMs. They present evidence that security is a top concern in the adoption of IDaaS.

Sherlock 2014 [8] discusses the concepts of Identity as a Service (IDaaS) and Federated Identity Management (FIM) and their acceptance among various institutions and the general population. FIM and IDaaS have been well received in educational, commercial, and government organizations but amongst the general population, it is a lot less accepted due to trust issues. One of the main hindrances to the adoption of FIM and IDaaS is the lack of understanding and trust among the general population [17]. Despite government efforts, notably in the European Community, to establish a

common credential provider and broker-based approach, there are still barriers to acceptance, such as concerns over the unauthorized release of private information and the lack of a clear definition of what constitutes an electronic identity (eID). The paper further emphasises the questions arising orientating around the truthfulness of IDaaS, such as risks of Man-in-the-Middle attacks, misuse of IdP and SP with user identity information, and the clear definition of liability arrangements are identified as potential risk areas.

In summary, while technological solutions for identity management, such as IDaaS have been proven to be viable and cost-effective, a full-scale implementation is currently constrained by trust issues among the general population [8].

Ducatel (2015) proposes a Horizontal Service design in turning Identity and Access Management (IAM) solutions into IDaaS, consisting of reusable, policy-driven feature enforcement to guarantee compliance integrity offered by a subscription-based service.[9] Bringing the concept of IDaaS closer to SaaS when it comes to business model.

The author describes trust and security as drivers of the adoption of cloud services, and that cloud service providers (CSPs) can demonstrate to customers (organisations or end consumers) how identity management is done: how is it protected, where is it stored, how do they mitigate identity-related risks. To that end, a classification framework for the level of mitigation for risks is offered. The need for transparency regarding exposure is important since "IDaaS for cloud services implies a fragmented model of identity responsibilities". In short, the author concludes that mitigating risk can not be done without transparency on the liability of each component in the system. Finally, the issue of storing identity is discussed, since this presents legal and regulatory ramifications. IDaaS customers should request and understand the Service Level Agreement (SLA) of the service to ensure it complies with internal policies and external regulations. This Horizontal Service architecture addresses trust-related issues for customers while being appropriate for multi-tier and multi-cloud deployments, it is argued, and therefore contributes to cloud adoption. Vo et al. 2016 [10], discuss three key roles in any cloud computing environment according to IBM's Cloud Computing Reference Architecture, namely service creator, cloud provider, and cloud consumers or end users. Three scenarios are showcased in these environments, including dynamic single-sign-on (SSO) issues where a user has to manage different credentials for different services within the same cloud provider, dynamic service binding where a service may need to partner with another service based on user behaviours, and identities roaming (users moving geographically and accessing services from different locations). The model described in the paper [10] discusses several components of an IDaaS model including:

- Policy Enforcement Point (PEP): This intercepts the authentication request and handles authorization for the service provider.
- Policy Decision Point (PDP): This is a mechanism for analyzing and deriving any elements related to Security Policy from an existing implementation.
- Policy Information Point (PIP): This provides user information for the PDP to make decisions and also handles identity roaming between IDaaS in different security domains.
- Policy Administration Point (PAP): This endpoint provides functionalities for operators of tenant applications to review the derived policies and configure them on demand [10].

The authors further explore identity roaming, adhering to privacy guidelines, and ensuring that user data is protected from unauthorised access or disclosure. The future work is an interesting suggestion: it is proposed to extend Topology and Orchestration Specification for Cloud Applications (TOSCA), a standard to describe topology for Cloud applications. The author suggests an extension of TOSCA to describe a model for IDaaS components, developing a mechanism to protect identity roaming against identity theft, and considering automated trust negotiation between IDaaS based on existing trust between mobile network operators [10].

In their following work, the authors present a novel approach to preserving privacy in IDaaS. In particular, they focus on how Personal Identifiable Information (PII) can be disseminated across multiple cloud services, via federated identity management, while still ensuring PII is not misused or accessed by unauthorised entities. The proposed technical solution combines Purpose-based Access Control (PBAC) and Attribute-based Encryption (ABE). [11]

They go into detail on the implementation of such system, including how authentication and encryption would work. In terms of adoption, they explain how this solution can support organisations

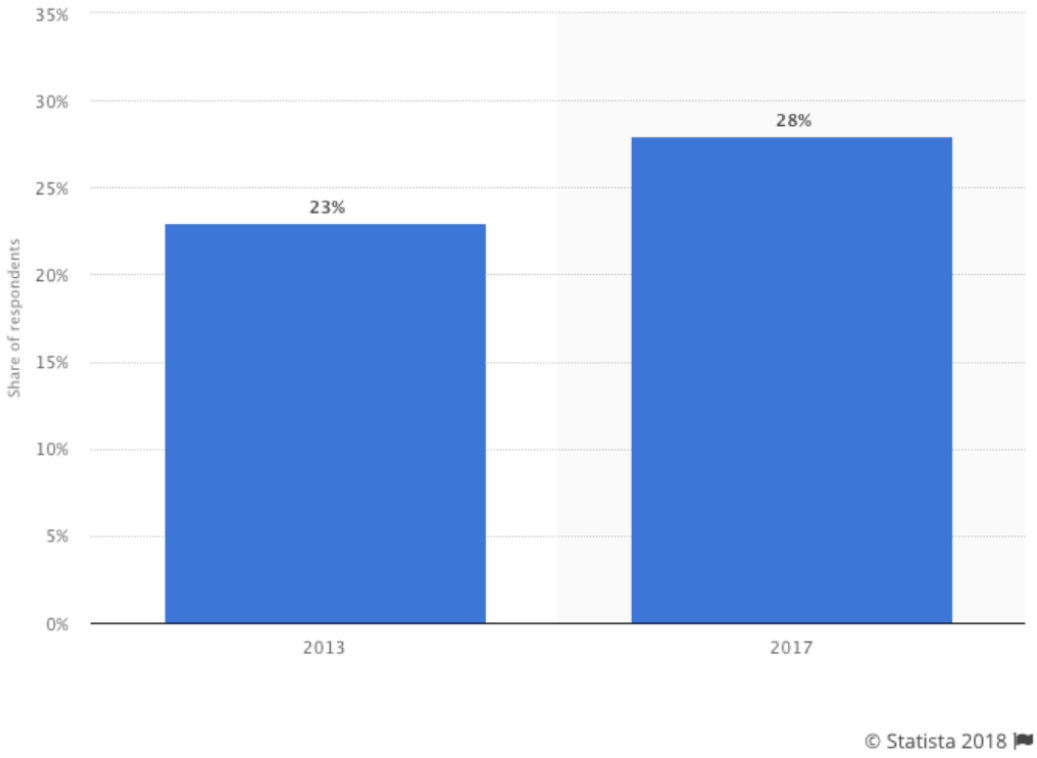


Figure 2: Graph showing increase of acceptance of IDaaS from 2013 to 2017 [16]

in keeping compliance: "'European Union Data Protection Directive' prohibits data transfer or processing by a service that is hosted in a country with a weak privacy protection law. By applying these guidelines, our access control model takes time, purpose, location, and domain as the main factors for describing the disclosure policy." Interestingly, this work focused on preserving user privacy in IDaaS for federated environments, whereas previous work focused more on the issue of security and trust. PII might have more specific regulations, such as the "European Union Data Protection Directive", and therefore it is valuable to design a system that is convenient for the end users and services in terms of its portability and usability but that accounts for the sensitivity of the information being disseminated.

In his work 'The rise and rise of ID as a Service' (2018), John Mears describes how IDaaS is changing how biometric matching and identity services are provided. [1] The author describes the ambiguity of the term IDaaS (in some contexts, only describing a cloud-based subscription service, in others as any identification function open to multiple entities, regardless if in the cloud or not). In any case, the author describes the main functionalities of IDaaS as: Authentication (with authorization included here): enables a service "to verify that a person asserting an identity is indeed the person they claim to be". Identification: "the comparison of an unknown person or user to a potentially large gallery of known subjects". [1] So far in this review, only the aspect of authentication (and authorization) was included, but not services for identification. The author states that "organisations have adopted a cloud-first or cloud-only strategy to ensure they can reap the benefits described in the NIST definition", namely: that cloud computing provides ubiquitous and convenient access to configurable computing resources, that can be easily and quickly provisioned and released.

In that sense, the financial and competitive drivers in adopting IDaaS are highlighted: the ready availability of these services and the flexibility to scale them as demand requires. Also, regulatory motives, such as data security and regulation specific to certain industries. IDaaS appear as a solution to these challenges. Three main types of organisations and industry sectors are listed as being attracted to IDaaS: government agencies, due to the elasticity and scalability of IDaaS systems; financial industry, due to regulatory, performance and security considerations; and the healthcare industry, to reduce errors and fraud. The paper also focuses on biometric modalities in IDaaS applications:

fingerprints; facial, iris and voice recognition; and other less common methods (e.g. DNA). With the mass adoption of mobile and other devices with biometric and behavioural authentication capabilities built-in, the acceptance of advanced authentication increases.

A hindrance to adoption is that “IT staff perceive there is a loss of control of data, and subjects worry about the security of their data being stored in the cloud”. Once more, concerns with security are a blocker, but the growth of SaaS (and IDaaS as its subset) suggests that the benefits previously mentioned are worth it. In this work, many specific cases and solutions are mentioned, and the author makes a logical case for the rise of IDaaS, linking it to the development of other technologies, such as biometrics in customer devices.

On their work in 2019, Vo et al. expand on their previous work of IDaaS as a way to preserve user privacy while offering interoperability in a federated system.[12] They emphasise that there is a demand for "highly secure and flexible access control for identity federation" but this security feature is not a core competency of service providers, leading cloud services to prefer outsourcing IAM to third parties. Another factor is that the human-PC (personal computer) link is the weakest link when it comes to identity protection, more so than the links between PC, service provider and identity provider, and therefore, reducing human interaction from identity disclosure is desirable.

The main content of this paper is the architecture design of their proposed system, which enables the preservation of user privacy, while providing identity propagation between services or intermediaries in a secure way (i.e. with PII encrypted in the identity provider). They offer a solution for many of the challenges in IDaaS adoption and disclaim the technical considerations in doing so.

Chau 2019 [13] focuses on exploring the factors that drive the adoption and usage of cloud services in organisations. Cloud computing provides an alternative model to traditional IT deployment and governance, enabling businesses to focus on their core competencies instead of managing large-scale IT infrastructures, such as users’ identities. This study looks into the decision-making process of whether or not to move an IT operation to the cloud. A survey was carried out, examining factors such as extrinsic motivation, intrinsic motivation, perceived risks, and resource constraints. Cloud services can be delivered through three models:

- Software as a Service (SaaS)
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)

and can be deployed using private, public, community, or hybrid infrastructures [13]. IDaaS specifically focuses on managing user identities and access control.

Similar to the aforementioned cloud services, IDaaS is influenced by the same factors such as extrinsic motivation, intrinsic motivation, perceived risks, and resource constraints. For example, a company might be motivated to use IDaaS due to its benefits of centralised identity management and reduced overhead costs (extrinsic motivation), or because it aligns with the company’s values of using modern, cloud-based technologies (intrinsic motivation). Chau 2019 [13], emphasises which factors can influence the potential adoption of IDaaS. Understanding these factors can help cloud service providers improve their offerings and address the concerns of potential customers, thus facilitating wider adoption of these technologies [13].

Gomma 2020 [14], paper focuses on the concept of IDaaS in the context of cloud computing. It acknowledges that while IDaaS brings many benefits, it also introduces security challenges, particularly regarding identity theft, as previously discussed. The paper proposes the use of Virtual Identity (VID) as a solution to enhance security within the IDaaS framework [14].

The paper explains that IDaaS is commonly used for authentication in Software as a Service (SaaS) cloud deployment models and can be provided by third-party identity providers. The authors propose the use of VID, which allows for anonymous Single Sign-On (SSO) in distributed cloud service environments. They design a VID creation framework using Elliptic Curve Cryptography (ECC) and implement two approaches:

- Identity Based Encryption (IBE)
- Pseudonym Based Encryption (PBE) using the MIRACL security library

To assess the security of their proposed solutions, the authors use the AVISPA tool, which analyzes the formal models of security protocols. The analysis shows that both the IBE and PBE protocols are secure with no vulnerabilities found. The authors conclude that their VID approaches based on IBE and PBE are suitable and scalable for securing anonymous communication in cloud services environments.

The contributions of the paper include the design and modelling of the VID framework, the implementation of the models using the MIRACL security environment, the validation of the protocols using AVISPA to assess their security measures, and the comparison of the proposed solutions with related work in terms of security and cost. It provides context and an indication of how IDaaS can be further developed and possibly be more generally accepted.

Future directions for the research include integrating the proposed VID solutions into a running cloud environment and further studying their implementation and performance in real-world scenarios [14].

Sharma et al. (2020) provide a systematic literature review on the topic of cloud computing adoption, using manual analysis and natural language processing (NLP). [3] It's an extensive work, ranging from describing which theoretical frameworks are utilised for studying cloud computing adoption (with the most common being TOE - Technology Organization Environment by Tornatzky et. al; and TAM - Technology Acceptance Model by Davis); the critical factors in the adoption of cloud computing; what are the significant causal relations of influencing factors are important; and if there are any differences in these factors, with respect to developed and developing countries.

In their review, the authors compiled and aggregated commonly mentioned keywords across 201 curated studies based on their criteria.

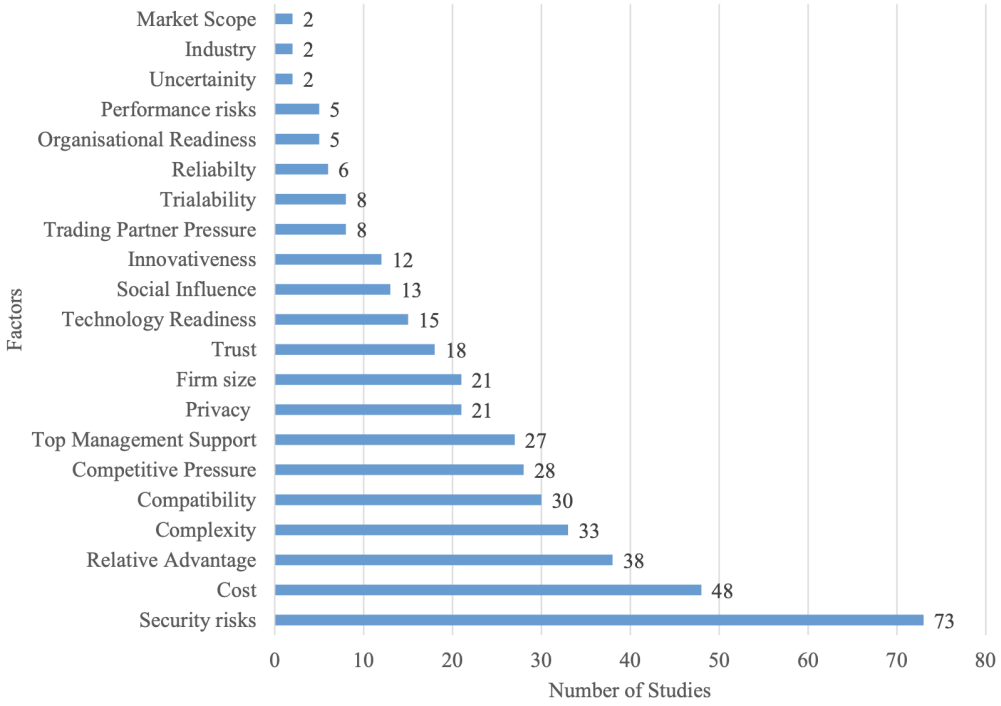


Figure 3: Factors influencing cloud computing adoption (worldwide) [3]

The most commonly cited words/phrases in the studies the authors reviewed are security, cost, trust, privacy, risk and virtualization. Figure 2 shows the keywords already aggregated in factors, with the five most commonly cited being security risks, cost, relative advantage, complexity and compatibility. Additionally, they uncovered a strong association between the terms security, privacy and lock-ins. Existing users of cloud services considered compatibility, security and technical support from the service provider as important factors.

An important finding in this work is that "most of the research examined specific aspects of the first-time adoption (...), but only a handful of researchers usually focus on which factors led to the continued urge and usage of adoption". And only one study mentioned factors that could lead to attrition of cloud services. These are areas in need of more research.

The landscape in the adoption of cloud, and hence how identity is managed in systems, has changed considerably in the last decades. Today, there is a great need for robust identity management, given the constant risk and massive negative impacts of breaches and attacks. At the same time, due to the multiplicity of users' devices, and applications and services on the internet, identity have to be easily manageable. IDaaS is an apt solution for this, and hence it is also known as "modern identity", possessing the capabilities of: directory (i.e. storage of data, metadata and policies on identity); single sign-on (SSO); multi-factor authentication (MFA); and provisioning and workflows (capabilities on managing the service, including automation).[18]

More recently, there are many providers for IDaaS as an enterprise product, with different capabilities. Organisations can evaluate them based on their needs and the capabilities of each of these products, such as single sign-on (SSO), multi-factor authentication (MFA) and type of authentication. [19] The size of the IDaaS market was valued at $5.5 billion dollars in 2021, and is projected to reach $41.9 billion dollars by the end of 2031, proving that the technology is being adopted at a high pace. [20]

4 Discussion and conclusion

As is the case for a novelty approach in a rapidly changing cloud computing ecosystem, the earlier works in identity management, specifically related to IDaaS, describe different possible architectural implementations. With the growth and adoption of cloud services from 2007 to today, a need for a scalable and robust approach to identity management is required, and IDaaS (with its many different possible implementations) is an apt solution for it.

The most commonly cited hindrance to the adoption of IDaaS is the issue of security and trust, even more so in its nascent stages. But with advancements in technology and many providers specialised in these solutions, it has matured and offers many advantages over developing and maintaining identity-management on-premises for organisations.

The main advantages of IDaaS closely resemble those commonly associated with cloud services: ready to use, with low cost of implementation, interoperability, easy to manage, elastic and flexible. However, externalising identity and access management can be perceived as a big risk for organisations, due to the lack of control.

These advantages propelled the adoption of IDaaS. Specifically to IDaaS: the possibility of managing identities in federated environments, reducing usability issues (such as the creation of multiple logins and passwords by users), and even security issues (with encryption, multi-factor authentication, and biometrics).

This literature review also brings attention to the fact that research on adoption (and attrition) factors specific to IDaaS is rare. For that reason, gaps had to be addressed by analysing factors related to cloud services adoption, and then putting them linking them to the context of identity management. Also worth noting is that the term IDaaS had different meanings from its inception (as a technical implementation) to now (fully-fledged enterprise products offered by third parties via cloud capabilities).

References

[1] Mears, J. (2018). The rise and rise of ID as a Service. Biometric Technology Today, 2018(2), 5–8. doi:10.1016/s0969-4765(18)30023-7 10.1016/s0969-4765(18)30023-7

[2] Emig, C., Brandt, F., Kreuzer, S., & Abeck, S. (2007). Identity as a Service – Towards a Service-Oriented Identity Management Architecture. Lecture Notes in Computer Science, 1–8. doi:10.1007/978-3-540-73530-$4_1$10.1007/$978 - 3 - 540 - 73530 - 4_1$

[3] Sharma, M., Gupta, R., & Acharya, P. (2020). Analysing the adoption of cloud computing service: a systematic literature review. Global Knowledge, Memory and Communication, 70(1/2), 114–153. doi:10.1108/gkmc-10-2019-0126 10.1108/GKMC-10-2019-0126

[4] Ates, M., Ravet, S., Ahmat, A. M., & Fayolle, J. (2011). An Identity-Centric Internet: Identity in the Cloud, Identity as a Service and Other Delights. 2011 Sixth International Conference on Availability, Reliability and Security. doi:10.1109/ares.2011.85 10.1109/ARES.2011.85

[5] Samlinson, E., & Usha, M. (2013). User-centric trust based identity as a service for federated cloud environment. 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT). doi:10.1109/icccnt.2013.6726636

[6] Stieninger, M., Nedbal, D., Wetzlinger, W., Wagner, G., & Erskine, M. A. (2014). Impacts on the Organizational Adoption of Cloud Computing: A Reconceptualization of Influencing Factors. Procedia Technology, 16, 85–93. doi:10.1016/j.protcy.2014.10.071 10.1016/j.protcy.2014.10.071

[7] Habiba, U., Masood, R., Shibli, M. A., & Niazi, M. A. (2014). Cloud identity management security issues solutions: a taxonomy. Complex Adaptive Systems Modeling, 2(1). doi:10.1186/s40294-014-0005-9 10.1186/s40294-014-0005-9

[8] Sherlock, J. (2014) Review of Barriers for Federated Identity Adoption for Users and Organizations. Available at: https://arxiv.org/ftp/arxiv/papers/1810/1810.06152.pdf (Accessed: 30 May 2023).

[9] Ducatel, G. (2015). Identity as a service: A cloud based common capability. 2015 IEEE Conference on Communications and Network Security (CNS). doi:10.1109/cns.2015.7346886 10.1109/CNS.2015.7346886

[10] Vo, T.H., Fuhrmann, W., & Fischer-Hellmann, K.P. (2016) Identity-as-a-Service (IDaaS): a Missing Gap for Moving Enterprise Applications in Inter-Cloud. Available at: https://www.cscan.org/openaccess/?id=306 (Accessed: 30 May 2023).

[11] Vo, T. H., Fuhrmann, W., & Fischer-Hellmann, K.-P. (2018). Privacy-preserving user identity in Identity-as-a-Service. 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). doi:10.1109/icin.2018.8401613 10.1109/ICIN.2018.8401613

[12] Vo, T. H., Fuhrmann, W., Fischer-Hellmann, K.-P., & Furnell, S. (2019). Identity-as-a-Service: An Adaptive Security Infrastructure and Privacy-Preserving User Identity for the Cloud Environment. Future Internet, 11(5), 116. doi:10.3390/fi11050116

[13] Agrawal, V. K., Agrawal, V. K., Taylor, A. R., & Chau, N. (2019). An Exploratory Study Of Factors Driving Decision Maker Intentions To Adopt Cloud Computing. Information Technology and Management Science, 22, 37–46. https://doi.org/10.7250/itms-2019-0006

[14] Gomaa, I., Abd-Elrahman, E., Hamdy, A., Saad, E. M. (2020). Automated Security Assessment for IDaaS Framework. Wireless Personal Communications. doi:10.1007/s11277-020-07860-8

[15] Ping identity (no date) Identity as a Service. Available at: https://www.pingidentity.com/en/resources/content-library/articles/identity-as-a-service-idaas.html (Accessed: 31 May 2023).

[16] Ani Petrosyan, Jul 7, 2022, Use of two-factor authentication among U.S. online users 2013-2017

[17] Roberts, B., & Chen, Y. (2023). Business Agility through Cloud-Based Services: An Examination of IDaaS Adoption. Journal of Information Systems Cloud Computing, 11(3), 256-269.

[18] Miller, L., & Hakamine F. (2020) Identity as a Service (IDaaS). John Wiley & Sons. Available at: https://www.okta.com/resources/whitepaper-identity-as-a-service-for-dummies/ (Accessed: 31 May 2023).

[19] Murphy, M. (2020) Comparing IDaaS (Identity-as-a-Service) Providers. Available at: https://jumpcloud.com/blog/comparing-identity-as-a-service-idaas-providers (Accessed: 31 May 2023).

[20] Transparency Market Research (2020). Identity-as-a-Service (IDaaS). Available at: https://www.transparencymarketresearch.com/identity-as-a-service-market.html (Accessed: 31 May 2023).

The Impact of Cloud Computing on High-Performance Computing, with a Focus on Performance, Latency, Security and Cost

Yihong Xi
Faculty of Science
14720035
University of Amsterdam
yihong.xi@student.uva.nl

Xingyou Li
Faculty of Science
14741628
University of Amsterdam
xingyou.li@student.uva.nl

Zhiheng Yang
Faculty of Science
14483262
University of Amsterdam
zhiheng.yang@student.uva.nl

Abstract

High-Performance Computing (HPC) has been traditionally executed on dedicated supercomputers, however, with the emergence of cloud computing, there's an increasing interest in leveraging its scalable and flexible infrastructure for HPC applications. This review focus on this intersection, including the performance, latency, data security, and cost issues associated with running HPC workloads in cloud environments. We outline the challenges that arise due to architectural differences between traditional supercomputers and cloud environments, the implications of latency on cloud workloads, the complex data security concerns, and the cost-effectiveness of utilising cloud resources for HPC. Despite the challenges, the flexibility and scalability offered by cloud environments present significant opportunities for HPC. This review thoroughly covers the challenges and current state of HPC in the cloud, providing a strong foundation for future research in this promising field.

1 Introduction

High-performance computing (HPC) is a method of processing large amounts of data and performing complex calculations at high speeds[1]. HPC has traditionally been associated with physical clusters of powerful computers, deployed and maintained by experts in buildings which are designed for cooling the whole system with best efficiency[2]. In traditional HPC architecture, multiple computers or computing units are networked together to form a cluster which contains much more powerful performance than one single computer[1]. These traditional, on-premise HPC clusters offer the advantage of dedicated resources, enabling high-speed computation and communication capabilities, which are especially beneficial for tasks with significant data transfer and synchronization needs.

HPC contributes to solving complex problems with reasonable time costs but is also a significant investment which is not affordable for normal consumers and small enterprise businesses, especially when they need to make a profit at first. However, the new emergence of cloud computing has

introduced a new paradigm for HPC. In that case, compared to the traditional situation, computing resources are provided as a service over the Internet, which eliminates the demand for building cooling systems, purchasing expensive computers and maintaining the running of the whole computing system.

As a result, HPC with the cloud can help users or organisations to save a significant number of money spent on building infrastructure and scale their computing needs dynamically according to their real-time workload requirement, offering advantages such as on-demand resource provisioning, cost flexibility, and ability to scale resources according to workload requirements.

While cloud-based HPC offers numerous benefits and has transformed the landscape of high-performance computing, it has not completely replaced traditional HPC. When we try to transform from the traditional cluster-based model to cloud-based HPC, are there some impacts happening or some issues arising? What cons and pros do cloud computing brings to the HPC? A thorough evaluation of its capabilities and limitations is necessary to address these questions and gain a comprehensive understanding of the impact of cloud-based HPC. To answer these questions, we try to evaluate cloud-based HPC in the aspects of performance, data security, latency and cost.

2 Performance

To dig into the performance of different, it is needed to figure out what impact the computing speed of HPC the most. According to Lisa Morgan's study [3], the speed of an HPC depends on its configuration, which means more clusters and cores enable faster enable parallel processing. Performance is also affected by the software that runs on the machine, including the operating system design and application design, and the complexity of the problem being solved. Another study mentions that the network speed and bandwidth are also significant for the computing speed of HPC[4]. As a result, to evaluate HPC's performance, we introduce two metrics: execution time and turnaround time.

2.1 Execution Time

Gupta[5] conducted a series of experiments on different computing environments, including supercomputers and clouds, to find the answers of what advantages the cloud-based HPC have, what kinds of applications are suitable for the cloud environment of HPC and how the applications can be used or optimized for cloud-based HPC. They explored OS-level containers, hypervisor- and application-level CPU affinity and their impact on performance.

The relative result shows in Figure1. In that experiment, benchmark applications including NPB[6], Jacobi2D, NAMD[7], ChaNGa[8], Sweep3D[9], NQueens[10]. The figure illustrates when the number of cores increases, the execution time of cluster-based HPC environments is less than cloud-based HPC environments due to InfiniBand network and better processors. However, in some applications such as NPB Embarrassingly parallel, coresJacobi2D and NAMD, the cloud-based HPC Open Cirrus can achieve similar performance with cluster-based HPC, probably due to those benchmark applications are not intensive on the communication between computing nodes. Nevertheless, in communication-intensive benchmark applications such as NPB IS and CHaNGa, cloud platforms may not be able to achieve optimal scalability due to bandwidth limitations. In contrast, cluster environments have infinite bandwidth, making them a better option for these applications.

The results of their study revealed that public clouds are cost-effective for small-scale applications and can be used in conjunction with supercomputers through techniques like cloud bursting and application-aware mapping. Besides, they also identified network latency as a significant limitation that affects the scalability of applications in cloud environments.

In another study, Gupta[11] conducted performance evaluations of HPC benchmarks on clusters, grids, and a private cloud, in which it was confirmed that HPC applications could experience performance degradation in the cloud if they heavily rely on communication. But if they are not communication-intensive, the performance of the cloud is relatively satisfactory, especially considering the favourable cost-performance ratio of the cloud environment.

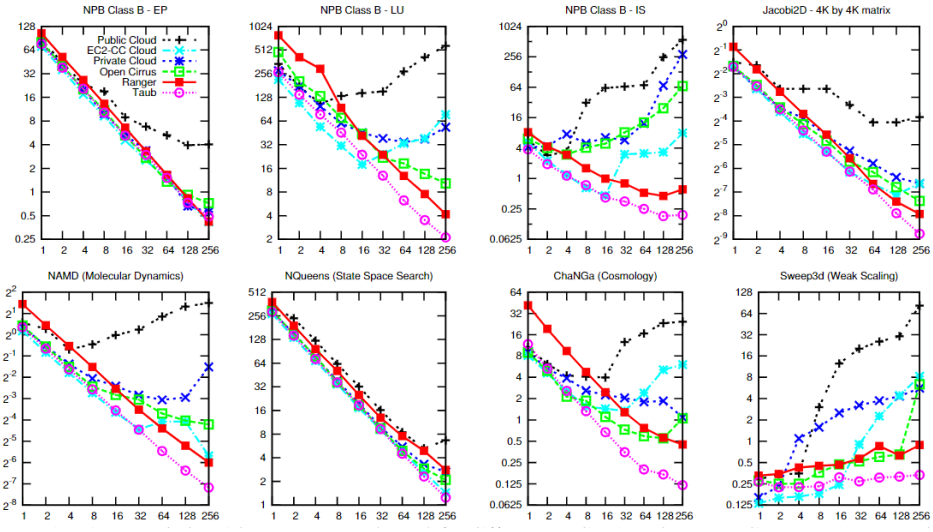


Figure 1: Time in seconds (y-axis) vs. core count (x-axis) for different applications (strong scaling except Sweep3D). All applications scale well on supercomputers and most scale moderately well on Open Cirrus. On clouds, some applications scale well (e.g., EP), some scale till a point whereas some do not scale[12].

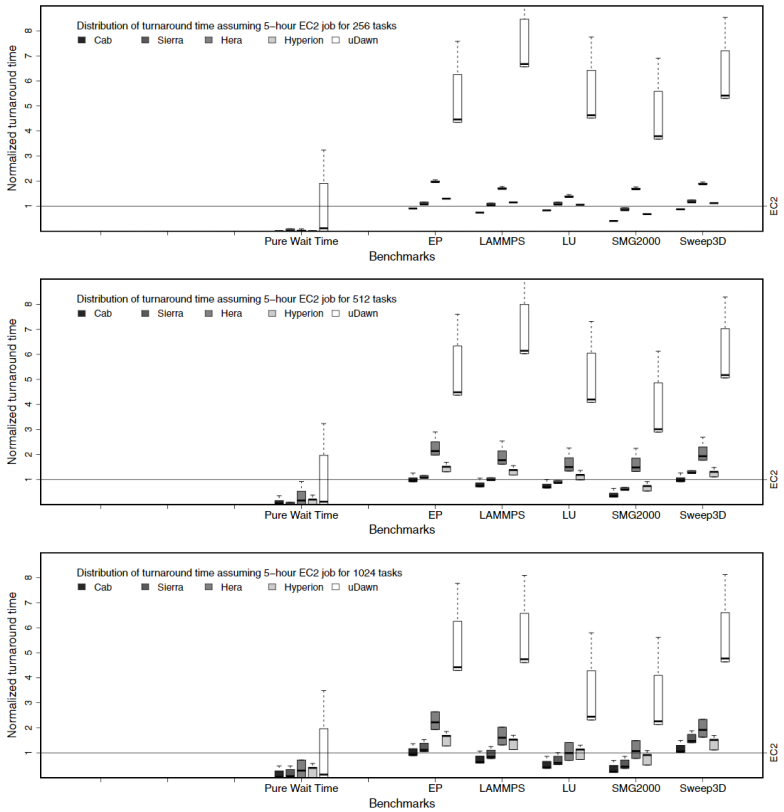


Figure 2: Comparison of total turnaround times on EC2 and LLNL clusters on 256, 512, and 1024 tasks. The figure shows turnaround times on LLNL clusters normalized to the EC2 turnaround times assuming 5-hour jobs on EC2. The y-axis represents multiples of EC2 turnaround time[13].

2.2 Turnaround Time

The study conducted by Marathe et al.[13] yielded intriguing results, highlighting the contrast between HPC clusters and EC2 clusters. The relative results are shown in Figure 2, which shows in many cases, the EC2 execution time is better at lower scales. When increasing the number of MPI tasks, higher-end LLNL clusters scale better than EC2, but it brings more queue wait time due to the demand for higher computation resources. Consequently, the turnaround time is longer in most clusters.

Although the HPC clusters exhibited superior raw performance, it was the EC2 clusters that achieved better turnaround times. In fact, the turnaround times for HPC on-premise resources were more than four times longer compared to their cloud counterparts, despite the local clusters executing tasks at a greatly faster pace. This discrepancy emphasizes the significance of considering turnaround time, a frequently overlooked but crucial factor in viability studies that focus on ensuring the quality of service (QoS)[13].

3 Latency

Traditional HPC systems have long been focused on optimising algorithms and memory management to achieve high computational performance. However, with the emergence of cloud-based HPC, new challenges related to latency have surfaced.

Latency refers to the delay or the time it takes for data or tasks to travel from a source to a destination within a cloud-based HPC system. It is influenced by various factors, including network infrastructure, data transfer protocols, scheduling schemes, and application characteristics. The network latency is a key limitation for the scalability of applications in the cloud[14].

3.1 Network latency

There are mainly two ways of big data stream processing: Continuous Data Streaming and Mini-batch Data Streaming. Continuous Data Streaming category includes Apache Flink streaming[15], Storm[16], and Twitter Heron[17]. The Mini-batch Data Streaming category includes Apache Spark streaming[15] and Dask Streamz[18], which also has better overall results (despite the stress of scheduling). While network latency is unavoidable due to limitations associated with the physical dimension. In HPC, where large-scale computations are performed on distributed systems, even small delays in network communication can have a significant impact on overall system performance[19]. High network latency can introduce bottlenecks and limit the ability of HPC systems to utilize computational resources effectively.

The impact of network latency on HPC applications is particularly pronounced in scenarios that require frequent communication and data exchange between computing nodes. For example, parallel applications that rely on message passing interfaces (MPI) heavily depend on low-latency and high-bandwidth network connections to achieve efficient inter-node communication[20]. Latency can directly affect the speed at which messages are sent and received, resulting in increased communication overhead and decreased parallel efficiency.

Various techniques have been employed in HPC systems to mitigate the effects of network latency. Network optimizations[21], such as reducing message size or employing efficient communication protocols, aims to minimize the impact of latency on application performance. Additionally, advancements in network technologies, such as high-speed interconnects and low-latency fabrics, have contributed to improving the overall performance of HPC systems by reducing network latency. In recent years, compared with traditional network architecture, another promising technology called Software-defined networking (SDN) has emerged as a potential solution[22] (shown in figure 3).

3.2 Data Transfer Protocols

TCP is widely recognized as the principal protocol for ensuring reliable data transfer in IP networks, demonstrating its effectiveness since its inception and remaining the preferred choice for most communication scenarios. Nevertheless, TCP's suitability is limited when it comes to latency-sensitive processing.

For example, in data centre environments, TCP latency can vary significantly. While best-case round-trip latency can be as low as 25 μs, latency outliers occur during congestion or link faults. These outliers range from 50 ms to several seconds, even with alternative network paths. Retransmission of lost TCP packets contributes to these outliers, as TCP implementations maintain high retransmission timeouts to accommodate potential delays within the operating system[23]. Thus, developers often prioritize optimizing other aspects, relegating TCP/IP to a secondary role. This becomes especially significant in streaming systems where the primary performance limitation stems from I/O constraints due to the need to transfer a large number of small messages[24].

Indeed, the existing protocols exhibit both strengths and weaknesses. However, achieving comprehensive optimisation of latency that meets the demanding requirements of HPC is a challenging task. Figure 4 explains some presented congestion control schemes, which also shows the lack of the ability to control latency[25].

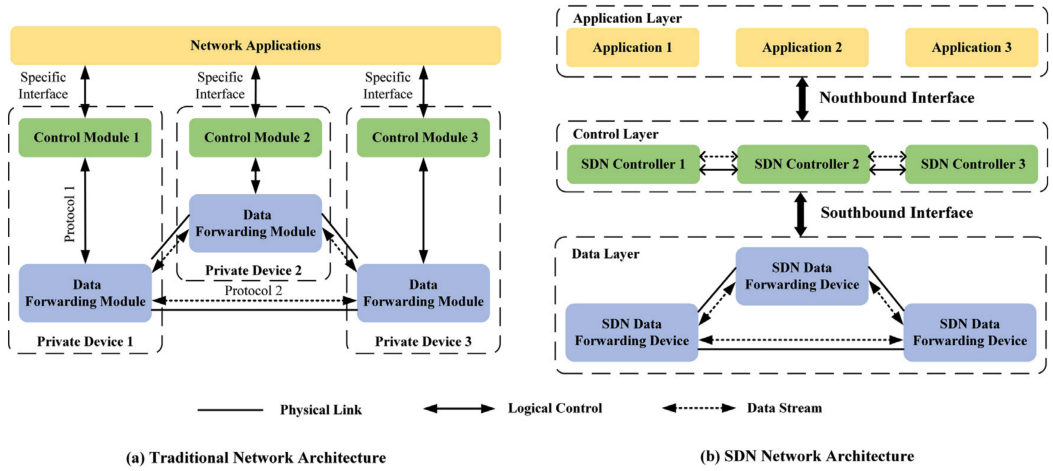


Figure 3: Comparison of traditional network structure and SDN network structure[22]

Type	Algorithm	Congestion control mechanism	Pros	Cons
Loss-based	NewReno	AIMD	Proven convergence, fairness	Inefficient in LFNs
	BIC	Binary search increase function	Higher efficiency	Too aggressive
	CUBIC	Cubic CWND function	RTT-independent fairness	High number of retransmissions
	Wave	Burst-based adaptation	Fairness and efficiency	Highly volatile RTT
Delay-based	Vegas	RTT changes as congestion signal	Fewer retransmissions, lower latency	Suppressed by loss-based flows
	Verus	Delay profile-based AIMD	Adaptability to volatile channels	High sender-side CPU load
	Nimbus	Explicit queue and cross traffic modeling	Scalable aggressiveness to deal with CUBIC	Unfair to Vegas and BBR
	LEDBAT	Extra delay to high-priority flows	Does not affect other flows	Limited to low-priority traffic
Capacity-based	Westwood	Bandwidth estimation to decrease CWND	Good in wireless and lossy links	No latency control
	Sprout	HMM capacity model	Low delay, customizable	Needs one buffer per flow
Hybrid	Compound	Sum of Reno and Vegas windows	Fast in LFNs, fair to CUBIC	No latency control
	Illinois	Delay used to determine CWND change	Fast in LFNs, fair to CUBIC	No latency control
	Veno	Explicit model of the buffer	Fast in LFNs, fair to CUBIC	No latency control
	BBR	Capacity and RTT measurement	High throughput, low delay	Fairness and mobility issues
Learning-based	Remy	Monte Carlo-based policy	Reaches capacity with low delay	Fairness issues with other TCPs
	TAO	Advancement on Remy	Fairness issues solved	Requires knowledge of the network
	PCC	Online experiments to determine CWND	Good in high RTT networks	Untested with bufferbloat
	TCP-RL	Reinforcement learning to determine CC algorithm	Self-organizing capabilities	Untested in highly dynamic environments
	QTCP	Reinforcement learning to select CWND	Higher throughput than NewReno	Limited performance evaluation

Figure 4: Main Presented Congestion Control Schemes[25]

3.3 Scheduling Latency

This refers to the delay in scheduling and executing tasks across distributed computing clusters due to resource contention and scheduling algorithm effects. Task scheduling involves allocating tasks to computing nodes to achieve efficient parallel computing. In a multi-node HPC system, the task

scheduler needs to consider factors such as node load, task dependencies, communication overhead, etc., to determine where and when tasks should be executed. This process takes time, resulting in increased waiting time for tasks and introducing task scheduling latency. In the functional model of schedulers, the architecture of every job scheduler comprises four essential functions: job lifecycle management, resource management, scheduling, and job execution[26], shown in figure 6.

As shown that scheduling latency is involved in all components because of the resource contention, complexity of scheduling algorithms, scheduling decisions, and dependencies between jobs. Currently, the industry is focusing on scheduler algorithms as hardware resources are not unlimited. There are a number of schedulers available, and they have been improved in terms of reducing latency, each with its own advantages. Figure 6 and Figure 7 show the results of a controlled experiment to measure scheduler latency[26].

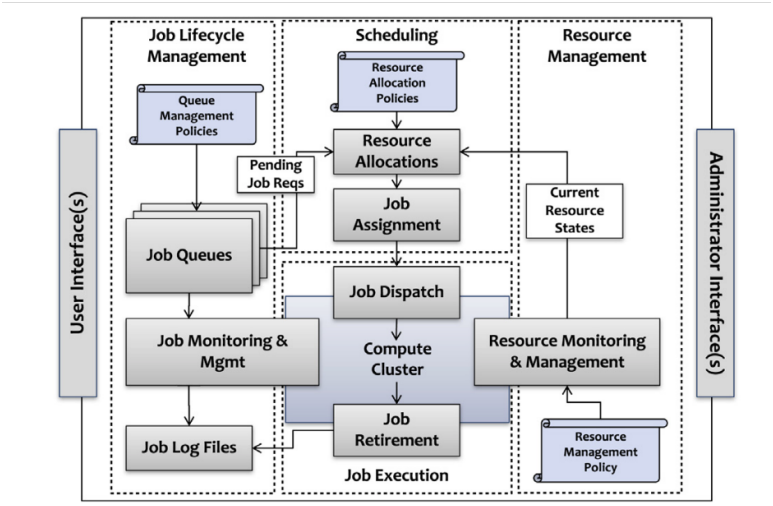


Figure 5: Key Components of Cluster Schedulers[26]

Configuration	Rapid Tasks	Fast tasks	Medium tasks	Long tasks
Task time t	1 s	5 s	30 s	60 s
Job time per processor T_{job}	240 s	240 s	240 s	240 s
Tasks per processor n	240	48	8	4
Processors P (cores)	1408	1408	1408	1408
Total tasks N	337,920	67,584	11,264	5632
Total processor time	93.7 h	93.7 h	93.7 h	93.7 h
Runtimes (sec)				
Slurm	2774, 2787, 2790	622, 603, 606	280, 278, 255	287, 264, 300
GE	3057, 3073, 3082	622, 634, 623	278, 279, 277	275, 281, 274
Mesos	1794, 1795, 1792	366, 364, 367	280, 280, 281	306, 306, 305
Hadoop YARN	–	2013, 1798, 1710	479, 472, 510	342, 445, 347

Figure 6: Comparison of Performance in Schedulers (Hadoop YARN Rapid Tasks failed executed)[26]

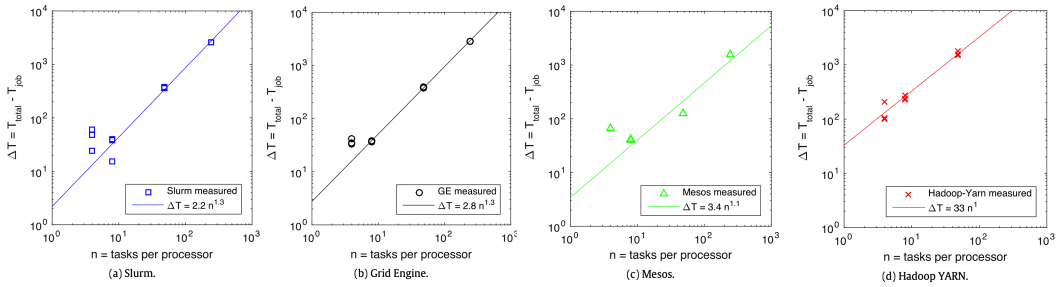


Figure 7: Comparison of Performance in Schedulers (tasks per processor)[26]

4 Data Security

In traditional HPC environments, data is often stored and processed on local servers or clusters, which means data transfer typically occurs over local networks, which can be fast but also limited by the capacity of the local infrastructure. In contrast, the advent of cloud computing allows data to be stored and processed on remote servers(data centres), which can be located anywhere in the world[27]. Thus, the data security problem is crucial. Wang et al.[28] introduced seven aspects of data security concerns, and this study discussed Data Destruction Security, Data Integrity Security and Data Sharing Security.

4.1 Data Destruction Security

In the cloud computing environment, after data simulation, to ensure data security or free storage capacity, users may delete some data. However, if the data is not destroyed immediately or completely, and even secretly stored, problems such as data leakage and illegal usage may occur[29]. Indeed, even if the user successfully removed the data from the cloud, some technologies can recover the deleted data from the hard disk[30]. It is possible to recover the data for memory locations not written with new contents[31]. Thus, assured deletion is needed. Deletion is assured when deleted data is irrecoverable or not accessible. To address this issue, Tang et al. proposed the FADE[32] system, which is a trusted third-party system that helps manage encryption keys for cloud data storage. It is designed to ensure that deleted data remains permanently inaccessible by making the encryption key unrecoverable once the data is deleted. Thus, as long as the user encrypts the data before uploading, and after deletion, if the malicious cloud provider could recover the data, they cannot access the data without the encryption key. However, if the cloud provider is not trusted, so is the third-party system.

Regarding this problem, Yang et al.[33] propose a new counting Bloom filter-based scheme for secure data transfer and deletion in cloud computing. The proposed scheme guarantees verifiable data transfer and deletion without requiring any trusted third party. When a deletion request is received, the cloud server checks whether the hash value of the requested item exists in CBF by performing a lookup operation. In most cloud setups, there are usually multiple versions of a file. Unfortunately, these approaches do not take that scenario into account. Rahumed et al. extended FADE to a new system FadeVersion[34], which supports both version control and assured deletion, each version of a file has its own encryption key. If this key is revoked, it can be assumed that the version of the file will be permanently deleted.

Nevertheless, Mo et al.[35] pointed out the FadeVersion creates a heavy burden on users because the volume of the keys can be huge if fine-grained deletion is required. They designed a new data structure called Recursively Encrypted Red-black Key tree (RERK). RERK assumes the responsibility of securing data and encryption keys stored in the cloud. The cloud user still maintains a master key or metadata, which helps to manipulate encryption keys and data stored by the provider. Fine-grained deletion is supported, but the issue of deleting multiple copies still needs to be addressed.

Maintaining accessibility and reliability of data in the cloud demands multiple replications. However, deleting this data can pose a challenge as it requires ensuring the complete removal of all copies. The traditional schemes are only aware of the existence of the copies but do not know the quantities and locations[36, 37]. Lai et al. present a scheme that uses the collision-resistant hash function and the key modulation algorithm to remove duplicate data and achieve fine-grained assured deletion without a third-party system[38].

4.2 Data Integrity Security

The data integrity assures there is no modification without users' knowledge, which is useful for the validity of data, and also promises the reliability and uniformity of data. Lack of integrity allows when intruders or malicious users gain the stored data, they can attack the data, including data modification attack, data leakage attack and Tag forgery attack. Some schemes[37, 39, 36, 40, 41] have been proposed, which are useful to ensure the storage correctness and do not need users to process the data, but these schemes are focus on a single server and most of them do not consider the dynamic data, they can not solve all security threats. Some complementary protocols[42, 43, 44] ensure storage correctness across multiple servers or peers. However, none of the above approaches involves dynamic data operations, since the cloud-stored data can be updated dynamically, it is a limitation

for the data storage. A new protocol has been suggested by Luo et al.[45], aimed at enhancing the checking of remote data possession. This innovative protocol leverages Hierarchical Linear Aggregation (HLAs) and RSA construction, substantially boosting data integrity. An added feature of the protocol is its public verifiability, increasing its adaptability and supporting data dynamics. He proved his protocol is security against server, and ensures file privacy against third-party auditors. This makes it an ideal solution for cloud storage systems.

Nevertheless, the schemes above do not take into account the computation capability of the users, while the verification cost is expensive. Zhang et al.'s scheme[46] is more efficient, which the overhead is independent of the number of verification tasks. It also supports dynamic operations, and he use performance analysis proved his scheme achieves lightweight verification.

4.3 Data Sharing Security

Since the data was stored in the data centres, it also introduces security problems. For example, the data centres can be attacked by malicious individual users or other cloud platforms. In this scenario, users can implement four kinds of common security mechanisms which are also suitable for the traditional HPC to protect the data, including file-level, database-level, media-level and application-level data security. However, Cheng et al.[47] pointed out that traditional security mechanisms may not be able to keep up with the volume and velocity of data being generated and processed in modern information systems. Additionally, some schemes that have been proposed to enhance the security of cloud client data in data centres mainly focus on designing algorithms to keep data confidential, which can be costly and inefficient when applied to big data. Cheng proposed a method that splits the data into parts, each of these parts is then encrypted using a cryptographic virtual mapping technique and uploaded to different cloud storage service providers. It addresses the efficiency problem of traditional security mechanisms by finding a better balance between the overhead of data partition and data uploading, ensuring the storage and sharing of confidential big data.

Besides, Razaque et al.[48] proposed a simplified approach, which divides the data into n parts, encrypts them and uploads them to multiple clouds, like the first step of Cheng's method[47]. The principle of this method is that the data was split defined by a polynomial function, only the user obtains at least k parts of the data with the same polynomial function, they can reconstruct the original data. In this strategy, the prerequisite of at least k data parts and the polynomial function secure the data at the same time. Apart from the procedure in the paper, we can improve the scheme by encrypting different sets of data parts with different public keys.

However, those schemes can only be considered to ensure the security problem of the individual data owner, but not for data sharing between several group members, which is common in current scientific research. Hence a protocol that supports secure group data sharing under cloud computing is needed. Such protocol is called key agreement protocol, which as first designed by Diffie-Hellman in their seminal paper[49], aims to generate a conference key for multi-user to ensure data sharing security. Diffie-Hellman key agreement can only support two parties, and it does not provide an authentication service which makes it vulnerable to attacks. Currently, many researchers have improved the key agreement and gained better variants, like Shen et al.[50] introduced the symmetric balanced incomplete block design (SBIBD) that supports group data sharing in cloud computing, which reduced the complexity of communication and computation of the key agreement protocol.

4.4 Challenges and Limits

Data encryption plays a significant role in cloud data security, and the overhead of emerging solutions is expensive, they need notable computation for encryption and verification, so the development of encryption-ready cloud applications could be an area to be explored. Trusted computing is such an alternative solution, cloud providers could offer secured hardware containers within their cloud infrastructures, but the secure hardware is expensive, and it requires secret key handover from the user to trusted hardware; the adaptability is also limits[51].

5 Cost

The movement of HPC workloads to the cloud has become a trend in recent years. Many businesses embark on this journey by conducting a total cost of ownership (TCO) analysis, assessing whether

cloud-based HPC solutions provide a more cost-effective alternative to traditional on-premise systems [52]. In this section, the cost of ownership of cluster-based HPC and the pricing methodology will be discussed. Then we will introduce a comparison of computing costs for running small jobs between cluster-based HPC and cloud-based HPC.

5.1 Cluster-based HPC Ownership Cost

It is a significantly large investment to build and own a cluster-based HPC infrastructure. Here's a list of typical direct costs of cluster-based HPC ownership[53]:

- Hardware: This category comprises the cost of physical servers, replacement parts, and associated materials.
- Security Measures: This includes tools like antivirus software.
- Software
- Storage
- Various Licenses
- User Support
- Off-site Backups

The list illustrates that to build an on-premise HPC system, people should consider amount of expenses. For example, SURF, the Dutch cooperative association in which educational and research institutions join forces, builds its supercomputer Snellius. According to the statistics, Snellius will give even more calculating power to scientific research in the Netherlands with a peak performance of 14 petaflops/s. However, the cost of Snellius is about 20 million euros which is unaffordable for most businesses[54].

5.2 Cloud HPC Pricing Methodology

Cloud-based HPC is more cost-effective for consumers due to flexible resource allocation, on-demand pricing, high resource utilisation, and versatile service options. This fee-based model, known as the pay-as-you-go approach, is friendly to persona users. Likewise, in dynamic markets, small and medium enterprises with expanding operations and an existing HPC infrastructure may hesitate to invest in on-premise resources and instead prefer adopting a pay-as-you-go model[5]. Leveraging diverse architectures with various interconnects, processor types and memory sizes enables improved resource utilisation on a global scale, surpassing the limited options available within a single organisation.

5.3 Cost Comparison

From Figure 8, the "Cray XC-40" is one instance of the cluster-based HPC system SahasraT[55], while the left columns are instances of HPC cloud platforms such as Amazon Web Server, Google Cloud Platform and Microsoft Azure. The cost of raw computation on the cloud is approximately 2.8 times higher than on SahasraT($4,836,888), while EC2 provide the best price($13,435,823). It should be mentioned that the cost comparison only considers the raw computing fee, regardless of some hidden expenses[53]:

- Tools used for temperature control in the data center
- The cost of set up, configuration, and ongoing upgrades
- Staff salaries for administrators that maintain an on-premise data center
- Networking infrastructure set up and ongoing maintenance
- Depreciation of the hardware and software
- The cost of keeping the servers powered 24/7

One possible reason why Cloud cost more than on-premise is that existing HPC centers are already consolidated and typically have high average utilisation, thus they are usually more cost effective[56].

Moreover, dedicated HPC systems usually provide performance benefits and essential services such as user support and training that are not available in commercial clouds, which can be thought as another cost benefit for consumers.

However, in the field of HPC, it should be emphasized that Cloud computing is cost-efficient for users with sporadic and varied compute resource needs, especially when the users' workload is relatively low. But for large computing job with a continuous workload, the pricing model pay-as-you-go would be considerably more expensive.

In conclusion, both cloud-based and on-premise HPC systems have their unique cost structures and advantages. The choice between the two should be driven by a careful analysis of an organisation's specific computational needs, financial capabilities, and long-term strategic objectives.

	Cray XC-40	AWS c4.8xlarge	Google n1-standard-32	Sabalcore	Azure Standard_F16s
Cost per physical core hour - 5 Year lifetime of Cray @10% AMC	$0.0126	$0.0350	$0.0540	$0.0700	$0.0620
Cost of 10 GB (Ingress + Egress + Storage)	$0.000	$0.023	$0.047	$0.006	$0.083
Cost of running a 48 core job for 24 hours (10GB of data)	$14.52	$40.34	$62.26	$80.65	$71.51
Cost of 100 GB (Ingress + Egress + Storage)	$0.000	$0.234	$0.467	$0.231	$1.083
Cost of running a 240 core job for 24 hours (100 GB of data)	$72.58	$201.83	$311.51	$403.43	$358.20
Cost of 1 TB (Ingress + Egress + Storage)	$0.000	$2.343	$4.667	$3.810	$8.035
Cost of running a 240 core job for 24 hours (1TB of data)	$72.58	$203.94	$315.71	$407.01	$365.16
Cost of 10 TB (Ingress + Egress + Storage)	$0.000	$23.433	$43.333	$39.600	$80.533
Cost of running a 240 core job for 24 hours (10TB of data)	$72.58	$225.03	$354.37	$442.80	$437.65
Total Number of Core Hours used by small jobs (less than 1200 cores) from April 2015 - December 2017 is 383.88 Million Core Hours					
Cost of running jobs worth 383.88 Million Core Hours (10TB of data)	$4,836,888	$13,435,823	$20,729,563	$26,871,640	$23,800,641

Figure 8: Cost of running small HPC jobs (cluster vs Cloud)[56]

6 Discussion and Conclusions

Our study focused on the complexities of moving High-Performance Computing (HPC) applications to the cloud, which presents both advantages and obstacles. The objective of the study is to assess the potential and feasibility of cloud-based HPC in four crucial dimensions: performance, latency, data security, and cost.

In terms of performance, while cloud platforms can bring shorter turnaround time for HPC applications, challenges related to architectural differences persist, requiring effective optimisation strategies. Latency in the cloud environment presents another challenge, necessitating advances in technology to ensure seamless communication between tasks, especially for applications that require real-time responses. Data security emerges as a significant concern, with data destruction, data integrity, and data sharing requiring careful consideration. While solutions have been proposed, further refinement is needed to meet the rigorous security demands of HPC workloads in the cloud. It's important to note that the cost-effectiveness of using HPC in the cloud is flexible and can vary according to scale of the project and its specific workload requirements while the cost of ownership on on-premise HPC is usually tremendous, which is unaffordable for most consumers and small businesses.

This study has contributed to a better understanding of the complexities associated with transitioning HPC to the cloud. Nevertheless, given the technological complexities involved, more research is needed to delve deeper into the intricacies of this transition and to develop robust strategies to address the identified challenges.

To improve cloud performance for HPC applications, future research should concentrate on improving bandwidth and hardware specification, optimising scheduling algorithms, strengthening data security schemes, and minimizing trusted computing costs. It's also important to examine the implications of cost in transitioning to the cloud, especially considering different application scales and usage scenar-

ios. Additionally, exploring how emerging technologies and advancements in cloud infrastructure can enhance HPC and bridge the gap between HPC and the cloud would result in better performance, security, and cost benefits.

References

[1] Hpc Architecture Explained. https://www.weka.io/learn/ai-ml/hpc-architecture/, sep 24 2021.

[2] High-performance computing data center cooling system energy efficiency.

[3] L. Morgan. High-Performance Computing | Datamation. https://www.datamation.com/data-center/high-performance-computing/, apr 5 2019.

[4] A. Abdullahi. 10 Networking Trends in High-Performance Computing | Enterprise Networking Planet. https://www.enterprisenetworkingplanet.com/data-center/networking-trends-high-performance-computing/, jul 20 2022.

[5] Abhishek Gupta, Laxmikant V Kale, Filippo Gioachin, Verdi March, Chun Hui Suen, Bu-Sung Lee, Paolo Faraboschi, Richard Kaufmann, and Dejan Milojicic. The who, what, why, and how of high performance computing in the cloud. In *2013 IEEE 5th international conference on cloud computing technology and science*, volume 1, pages 306–314. IEEE, 2013.

[6] Nas Parallel Benchmarks. https://www.nas.nasa.gov/software/npb.html, mar 23 2023.

[7] Abhinav Bhatele, Sameer Kumar, Chao Mei, James C Phillips, Gengbin Zheng, and Laxmikant V Kale. Overcoming scaling challenges in biomolecular simulations across multiple platforms. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–12. IEEE, 2008.

[8] Pritish Jetley, Filippo Gioachin, Celso Mendes, Laxmikant V Kale, and Thomas Quinn. Massively parallel cosmological simulations with changa. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–12. IEEE, 2008.

[9] Chunye Gong, Jie Liu, Haitao Chen, Jing Xie, and Zhenghu Gong. Accelerating the sweep3d for a graphic processor unit. *Journal of Information Processing Systems*, 7(1):63–74, 2011.

[10] Vikas Aggarwal, Ian A Troxel, and Alan D George. Design and analysis of parallel n-queens on reconfigurable hardware with handel-c and mpi. In *2004 MAPLD Intl. Conference*. Citeseer, 2004.

[11] Abhishek Gupta, Paolo Faraboschi, Filippo Gioachin, Laxmikant V Kale, Richard Kaufmann, Bu-Sung Lee, Verdi March, Dejan Milojicic, and Chun Hui Suen. Evaluating and improving the performance and scheduling of hpc applications in cloud. *IEEE Transactions on Cloud Computing*, 4(3):307–321, 2014.

[12] A. Gupta, P. Faraboschi, F. Gioachin, L. V. Kale, R. Kaufmann, B. S. Lee, V. March, D. Milojicic, and C. H. Suen. Evaluating and Improving the Performance and Scheduling of HPC Applications in Cloud. *IEEE Transactions on Cloud Computing*, 4(3):307–321, jul 1 2016.

[13] Aniruddha Marathe, Rachel Harris, David K Lowenthal, Bronis R De Supinski, Barry Rountree, Martin Schulz, and Xin Yuan. A comparative study of high-performance computing on the cloud. In *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, pages 239–250, 2013.

[14] Marco AS Netto, Rodrigo N Calheiros, Eduardo R Rodrigues, Renato LF Cunha, and Rajkumar Buyya. Hpc cloud for scientific and business applications: taxonomy, vision, and research challenges. *ACM Computing Surveys (CSUR)*, 51(1):1–29, 2018.

[15] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering*, 38(4), 2015.

[16] Jan Sipke Van Der Veen, Bram Van Der Waaij, Elena Lazovik, Wilco Wijbrandi, and Robert J Meijer. Dynamically scaling apache storm for the analysis of streaming data. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 154–161. IEEE, 2015.

[17] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M Patel, Karthik Ramasamy, and Siddarth Taneja. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*, pages 239–250, 2015.

[18] Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th python in science conference*, volume 130, page 136. SciPy Austin, TX, 2015.

[19] Robert Underwood, Jason Anderson, and Amy Apon. Measuring network latency variation impacts to high performance computing application performance. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 68–79, 2018.

[20] Hugo A López, Eduardo RB Marques, Francisco Martins, Nicholas Ng, César Santos, Vasco Thudichum Vasconcelos, and Nobuko Yoshida. Protocol-based verification of message-passing parallel programs. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 280–298, 2015.

[21] Bob Briscoe, Anna Brunstrom, Andreas Petlund, David Hayes, David Ros, Jyh Tsang, Stein Gjessing, Gorry Fairhurst, Carsten Griwodz, and Michael Welzl. Reducing internet latency: A survey of techniques and their merits. *IEEE Communications Surveys & Tutorials*, 18(3):2149–2196, 2014.

[22] Binghao Yan, Qinrang Liu, JianLiang Shen, Dong Liang, Bo Zhao, and Ling Ouyang. A survey of low-latency transmission strategies in software defined networking. *Computer Science Review*, 40:100386, 2021.

[23] Leah Shalev, Hani Ayoub, Nafea Bshara, and Erez Sabbag. A cloud-optimized transport protocol for elastic and scalable hpc. *IEEE Micro*, 40(6):67–73, 2020.

[24] Pierre Matri and Robert Ross. Neon: Low-latency streaming pipelines for hpc. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 698–707. IEEE, 2021.

[25] Michele Polese, Federico Chiariotti, Elia Bonetto, Filippo Rigotto, Andrea Zanella, and Michele Zorzi. A survey on recent advances in transport layer protocols. *IEEE Communications Surveys & Tutorials*, 21(4):3584–3608, 2019.

[26] Albert Reuther, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa, et al. Scalable system scheduling for hpc and big data. *Journal of Parallel and Distributed Computing*, 111:76–92, 2018.

[27] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 grid computing environments workshop*, pages 1–10. Ieee, 2008.

[28] Fengling Wang, Han Wang, and Liang Xue. Research on data security in big data cloud computing environment. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 1446–1450. IEEE, 2021.

[29] Qingjie Liu, Xiaoying Wang, and Zhian Pan. Development and application of massive unstructured big data retrieval technology based on cloud computing platform. *Journal of Intelligent & Fuzzy Systems*, 38(2):1329–1337, 2020.

[30] Yunchuan Sun, Junsheng Zhang, Yongping Xiong, and Guangyu Zhu. Data security and privacy in cloud computing. *International Journal of Distributed Sensor Networks*, 10(7):190903, 2014.

[31] Jayachander Surbiryala and Chunming Rong. Data recovery in cloud using forensic tools. In *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 309–314. IEEE, 2018.

[32] Yang Tang, Patrick PC Lee, John CS Lui, and Radia Perlman. Fade: Secure overlay cloud storage with file assured deletion. In *Security and Privacy in Communication Networks: 6th Iternational ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings 6*, pages 380–397. Springer, 2010.

[33] Changsong Yang, Xiaoling Tao, Feng Zhao, and Yong Wang. Secure data transfer and deletion from counting bloom filter in cloud computing. *Chinese Journal of Electronics*, 29(2):273–280, 2020.

[34] Arthur Rahumed, Henry CH Chen, Yang Tang, Patrick PC Lee, and John CS Lui. A secure cloud backup system with assured deletion and version control. In *2011 40th International Conference on Parallel Processing Workshops*, pages 160–167. IEEE, 2011.

[35] Zhen Mo, Qingjun Xiao, Yian Zhou, and Shigang Chen. On deletion of outsourced data in cloud computing. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 344–351. IEEE, 2014.

[36] Kevin D Bowers, Ari Juels, and Alina Oprea. Proofs of retrievability: Theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 43–54, 2009.

[37] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597, 2007.

[38] Junzuo Lai, Jie Xiong, Chuansheng Wang, Guangzheng Wu, and Yanling Li. A secure cloud backup system with deduplication and assured deletion. In *Provable Security: 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings 11*, pages 74–83. Springer, 2017.

[39] Hovav Shacham and Brent Waters. Compact proofs of retrievability. *Journal of cryptology*, 26(3):442–483, 2013.

[40] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609, 2007.

[41] Giuseppe Ateniese, Roberto Di Pietro, Luigi V Mancini, and Gene Tsudik. Scalable and efficient provable data possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, pages 1–10, 2008.

[42] Thomas SJ Schwarz and Ethan L Miller. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 12–12. IEEE, 2006.

[43] Mark Lillibridge, Sameh Elnikety, Andrew Birrell, Michael Burrows, and Michael Isard. A cooperative internet backup scheme. In *USENIX Annual Technical Conference, General Track*, volume 2003, pages 29–41, 2003.

[44] P Ramesh Naidu, N Guruprasad, and V Dankan Gowda. A high-availability and integrity layer for cloud storage, cloud computing security: from single to multi-clouds. In *Journal of Physics: Conference Series*, volume 1921, page 012072. IOP Publishing, 2021.

[45] Wenjun Luo and Guojing Bai. Ensuring the data integrity in cloud data storage. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 240–243. IEEE, 2011.

[46] Yuan Zhang, Chunxiang Xu, Xiaohui Liang, Hongwei Li, Yi Mu, and Xiaojun Zhang. Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. *IEEE Transactions on Information Forensics and Security*, 12(3):676–688, 2016.

[47] Hongbing Cheng, Chunming Rong, Kai Hwang, Weihong Wang, and Yanyan Li. Secure big data storage and sharing scheme for cloud tenants. *China Communications*, 12(6):106–115, 2015.

[48] Abdul Razaque, Saty Siva Varma Nadimpalli, Suharsha Vommina, Dinesh Kumar Atukuri, Dammannagari Nayani Reddy, Poojitha Anne, Divya Vegi, and Vamsee Sai Malllapu. Secure data sharing in multi-clouds. In *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pages 1909–1913. IEEE, 2016.

[49] Whitfield Diffie and Martin E Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390. 2022.

[50] Jian Shen, Tianqi Zhou, Debiao He, Yuexin Zhang, Xingming Sun, and Yang Xiang. Block design-based key agreement for group data sharing in cloud computing. *IEEE Transactions on Dependable and Secure Computing*, 16(6):996–1010, 2017.

[51] Mahmoud Barhamgi, Arosha K Bandara, Yijun Yu, Khalid Belhajjame, and Bashar Nuseibeh. Protecting privacy in the cloud: Current practices, future directions. *Computer*, 49(2):68–72, 2016.

[52] W. Gentzsch. Squashing Total Cost Rumors of In-House vs. Cloud Computing. https://blog.theubercloud.com/squashing-total-cost-rumors-of-in-house-vs.-cloud-computing, jan 12 2021.

[53] Calculating On-Premise vs Cloud Costs — PMsquare. https://pmsquare.com/analytics-blog/2022/6/9/calculating-on-prem-vs-cloud-costs, jun 9 2022.

[54] Netherlands to build new national supercomputer | Computer Weekly. https://www.computerweekly.com/news/252500314/Netherlands-to-build-new-national-supercomputer.

[55] B. Joshi. Iisc's supercomputer SahasraT adds to Indian computing might. https://economictimes.indiatimes.com/news/science/iiscs-supercomputer-sahasrat-adds-to-indian-computing-might/articleshow/47380491.cms.

[56] Akhila Prabhakaran and J Lakshmi. Cost-benefit analysis of public clouds for offloading in-house hpc jobs. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 57–64. IEEE, 2018.

Application of Machine Learning algorithms in Cloud resource scheduling - Literature Study

Lixiang Zhang
Department of Computer Science
Universiteit van Amsterdam
14230054@uva.nl

Dingran Qi
Department of Computer Science
Universiteit van Amsterdam
14728575@uva.nl

Qingxian Lu
Department of Computer Science
Universiteit van Amsterdam
14723697@uva.nl

Abstract

Resource scheduling is one of the important tasks in a cloud computing environment that involves assigning different types of tasks and workloads to the available computing resources. Traditional static scheduling algorithms have become difficult to cope with these challenges, so the introduction of machine learning algorithms has become an effective way to solve the resource scheduling problem. In this article, we discussed the current investigation and application of machine learning algorithms in cloud resource scheduling and analyze the future trends of the application of Machine Learning in cloud resource scheduling.

1 Introduction

This article aims to give an overview of the literature regarding the application of Machine Learning and Deep Learning algorithms in a specific field of Cloud Resource and Monitor, Cloud resource rescheduling. The studies have been categorized into three sections, namely Machine Learning, Deep Learning, and Reinforcement Learning.

This paper aims to answer the following questions: What can Machine Learning do in the field of cloud resource scheduling? How does the application of ML improve the performance of resource scheduling in terms of QoS and SLA? What are the future trends? The research in this area covers different types of studies, ranging from theory to quantitative.

1.1 Cloud resource scheduling

Cloud resource scheduling is a complex concept. Cloud resources refer to various computing resources and services provided in the cloud computing environment, including virtual machines, storage space, bandwidth, databases, etc. Cloud resources are highly scalable and flexible, and users can elastically acquire and release resources according to their needs and pay on demand. [20]

Cloud resource scheduling refers to the process of efficiently allocating and managing available resources in a cloud computing environment. Due to the limited and diverse nature of resources in a cloud environment, the design of scheduling algorithms and policies is critical to achieve high performance, efficiency, and fairness.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1.1.1 Objectives

The goal of cloud resource scheduling is to achieve optimization in the following areas:

1. Performance optimization: Allocate resources to different tasks or services in a reasonable manner to maximize overall system performance and response time.
2. Resource utilization optimization: Avoid resource waste and over-allocation by dynamically adjusting resource allocation.
3. Load balancing: Balance the load of each node or server in the cloud environment to improve the reliability and scalability of the system.
4. QoS assurance: Ensure the quality of cloud services, such as response time, usability, and throughput, according to user needs and contractual requirements.

1.1.2 Auto-scaling

Elasticity is a prominent feature of cloud computing that allows for the dynamic acquisition and release of computing resources in response to resource requests within a cloud system. This capability empowers cloud service providers to deliver a good user experience by meeting the QoS requirements with high efficiency.

To effectively harness the benefits of elasticity, manual management of this dynamic process is not practical. Instead, adopting an automated and timely resource scheduling approach is crucial[13]. This approach is well known as auto scaling. By automatically scaling resources up or down, auto-scaling addresses the problems of over-provisioning and under-provisioning, optimizes resource utilization, better meets QoS requirements, and reduces SLA violations.

1.2 Machine Learning, Deep Learning and Reinforcement Learning

ML is a subset of artificial intelligence that focuses on developing computer systems capable of learning and improving their performance without explicit programming. ML algorithms enable machines to carry out tasks such as prediction, classification, and clustering by learning from data and recognizing patterns [11].

DL is a specialized branch within machine learning that specifically emulates the structure and functionality of neural networks in the human brain. By utilizing multi-level neural network models, DL enables computers to learn and process information like human brain. Deep learning models are consisted of multi-layer neural networks, including input layer, hidden layer, and output layer. The models are usually trained with large scale data and optimized by back propagation algorithm.

Reinforcement learning is another branch of machine learning, aside from supervised and unsupervised learning. It takes inspiration from the way humans learn through trial and error. In reinforcement learning, computer agents interact with an environment, taking actions and receiving rewards or penalties in return. By maximizing the cumulative rewards or minimizing penalties, these agents learn an optimal policy that determines which actions to take in different situations.

Combining Deep Learning and Reinforcement Learning, Deep Reinforcement Learning, DRL is a new methodology for solving problems when intelligent systems make decisions and learn in complex environments. In reinforcement learning, an agent learns optimal strategies by interacting with the environment to maximize cumulative rewards. It learns optimal strategies by taking different actions and observing the feedback from the environment. DRL uses deep neural networks as function approximators to learn the complex mapping relationships between states and actions. Deep neural networks can automatically learn abstract feature representations and train them end-to-end on large-scale datasets.[5]

2 Application of ML algorithm in Cloud Resource Scheduling

2.1 Resource Provisioning Through Machine Learning in Cloud Services

By combining machine learning for workload prediction and queuing theory for resource allocation, Mahendra, Rohit and Dharmendra proposed a framework optimizes system performance and ensures

efficient resource utilization with less waste of resource[17]. This study focuses on the task of resource provisioning, specifically aiming to find the best model for predicting traffic load on servers and estimating the required number of computing resources.

Workload prediction plays a significant role in auto-scaling, which refers to the ability to scale the resource up or down according to the incoming traffic load. It ensures efficient resource provisioning and helps implement service elasticity, reduce power consumption, and improve the quality of service (QoS). To get to know the incoming work load in the future, a machine learning-based auto-scaling mechanism is employed. This study investigates different models for workload prediction, including auto-regressive models, support vector machines (SVM), and long short-term memory (LSTM) networks. The data used for this study is time series data. Auto-regressive models, such as autoregressive (AR) and moving average (ARIMA), provide a linear estimation of server load based on historical data. However, they fail to capture the linear features of incoming workload patterns. In contrast, machine learning-based models can learn both the linear and non-linear patterns in incoming traffic loads, thereby improving prediction accuracy. SVM models with a radial basis function (RBF) kernel are one such example. Additionally, neural network models, particularly LSTM networks, are effective in predict workload in the future. For resource allocation decisions, queuing theory-based models are used. The M|M|C queuing model helps determine the most efficient allocation of resources. Performance metrics such as the average number of requests, average task response time, and throughput are used to evaluate the system's performance.

Sr. no.	Model name	MAE	RMSE	SLA Violations (%)	Unserved requests (%)	Over-provisioned	Under-provisioned
1	AR	0.1012	0.1279	45.2	5.1	39	– 82
2	MA	0.1006	0.1230	41.8	5.1	80	– 53
3	ARIMA	0.1655	0.2013	42.0	5.0	64	– 145
4	SVR(I)	0.0939	0.1131	43.1	4.9	47	– 93
5	SVR(II)	0.1017	0.1187	43.9	4.8	47	– 80
6	SVR(III)	0.1330	0.1532	44.2	4.5	37	– 92
7	LSTM-ReLu	0.0715	0.930	40.2	4.0	23	– 164
8	LSTM-sigmoid	0.0768	0.0986	40.9	4.1	34	– 194
9	LSTM-tanh	0.0663	0.0856	41.0	4.0	46	– 119

Figure 1: Summary table showing the benefits of LSTM over other methods

In conclusion, the authors proposed a machine learning-based auto-scaling framework that optimizes resource allocation. As figrue 1 summarized, LSTM is the best model for executing this task with the lowest over-povisioning rate and SLA violation percentage. By using an LSTM forecasting model, the researchers accurately predict future workload. Subsequently, the queuing theory-based M|M|C model is employed to efficiently allocate resources. Allocate resources to meet the estimated workload demand can avoid SLA violation and over-provisioning at the same time.

2.2 Efficient resource provisioning for elastic Cloud services based on machine learning

This paper proposed an efficient solution to provisioning of resources for Cloud services using machine learning techniques[9]. It is similar to last work but places greater emphasis on comparing Support Vector Machines (SVM) with different kernels and parameters to identify the best SVM model for workload prediction.

The proposed method in this study involves using time series data to forecast the traffic load over server. This method builds forecasting models based on past observations of the server load, enabling accurate predictions of workload in the future by Utilizing machine learning to capture underlying in the data. By leveraging the M|M|C model, the optimal number of resources required to satisfy the predicted traffic load can be estimated. This approach aims to optimize the efficiency of resource scheduling by reduce service response time and over-provisioning.

Compared to simple linear models, machine learning models, particularly SVM, demonstrate superior performance in capturing non-linear behavior patterns. Compared last work we discussed about, which claims neural network-based (NN) model is the best model for workload predicting, this paper believes that NN is deficient in being trapped in multiple local minima due to empirical risk minimization (ERM) rules. Meanwhile, SVM can obtain a unique and globally optimal solution based on structural risk minimization (SRM) [2]. The paper trains SVM models with different

kernels (polynomial, normalized polynomial, and RBF) and parameters. Evaluation of the models is conducted using measurements like MAE and RMSE, demonstrating that SVM models outperform simple linear models in terms of prediction accuracy. Based on the prediction results, resources are allocated using the queuing-theory method M|M|C and evaluated by performance metrics such as system utilization factor, average queue time, and average response time.

The result of this work shows that the resource management framework (SVM+M|M|C) accurately predicts the incoming server load, and efficiently estimates and allocate the required number of resources, reducing service time and fulfilling the SLA contracted by the user, as evidenced by significantly lower SLA violations.

2.3 VM Reservation Plan Adaptation Using Machine Learning in Cloud Computing

To achieve elasticity in cloud services, instead of using machine learning to predict workload and allocate resources accordingly, Sniezynski takes a different approach by considering a resource reservation-based auto-scaling method[14].

Resource reservation is an effective approach to mitigate the waste of energy caused by low resource utilization in cloud services. It involves reserving cloud capacity within specific time windows[18]. The use cases of resource reservation can be categorized into two main types: immediate use and future use. For immediate use, the challenge lies in the latency period between the request for resources and their allocation. During this period, the system must ensure the validity of the reserved resources, which will be utilized immediately. Meanwhile, for future use, resource reservation is aimed at preventing request failures due to predicted factors like traffic congestion or insufficient resource capacity.

```
1 begin
2 |  Download the plan P used and monitoring data R;
3 |  Train the model M_r on R;
4 |  Update the plan P – correct the number of machines in the plan
  |   according to the knowledge M_r;
5 end
```

Figure 2: Adaption based on machine learing

This work focuses on the second type of use case, where the authors define the reservation time window based on the starting time of the reserved resources. They aim to adapt the reservation plan using machine learning techniques. The reservation plan refers to a dynamic quota on the number of virtual machines that ensures an optimal balance. To achieve adaptation, the system monitors the capacity, such as CPU usage, and utilizes this data to predict the best reservation plan through machine learning. The current plan is then compared with the predicted plan and adjusted accordingly to align with the current system capacity as the process stated in figure2. Different machine learning models, including the Neural Network, linear regression, RepTree, and M5P, are applied for prediction. The evaluation of these models considers factors such as model learning time, prediction accuracy, and the performance metric Q, which measures the extent to which the number of reserved VMs deviates from the CPU limits. Among the models tested, RepTree persent a generally good performance with fast learning and relatively accurate results. However, the Neural Network outperforms RepTree in terms of resource utilization, as evidenced by significantly lower Q.

3 Application of DL algorithm in Cloud Resource Scheduling

3.1 DEARS:Deep Learning based Elastic and Automatic Resource Scheduling framework for cloud applications

Muhammad , Chen and Yutong Liu proposed an innovative DL Virtual Machines arrangement algorithm named DEARS to reduce the service latency and save energy and costs, thus improve the performance and efficiency of cloud applications.[10]

Basically, the DEARS model applies LSTM models to predict the resource demands of cloud applications based on the observed historical workload data of cloud servers. The structure of this model can be seen in figure3. It is made up of 5 modules, including:

- Workload Prediction Module: Get the input of workload trace and accordingly forecast the number of requests for the following time stamp
- Restriction Assessing Module: Calculate the required CPU cycles based on the number of requests passed in and assess the average utilization of virtual machines and physical machines.
- Virtual Resource Provision Module: Compare the current utilization status of VMs and the workload requirements
- Dynamic Consolidation Module: Screen physical machines, select and map virtual machines
- SLAs Feedback Module: Supervise the QoS and SLAs

For the LSTM model, at a specific time stamp t, the number of requests and responses are separately Nt and Rt. The input transformation works as follows: arrange N_t, R_t, t as the label of $N\_t - 1, R\_t - 1, t - 1$. The model consists of one 50-neuron input layer, a 250-unit hidden layer, and one fully connected hidden layer as an output layer with 2 units. They applied a dropout mask of 0.2 to the output of each LSTM as well. Regarding the critical metric for resource allocation in the modules, namely the calculation of the utilization, the authors calculated the utilization of virtual machines as follows:

$$Utilization_{avg} = \frac{CPUCyclesRequired + \alpha \cdot SLAs}{t. \sum NumberOfCores_i \cdot fi} \tag{1}$$

The numerator needs to calculate the required CPU cycles from the workload prediction and the SLA feedback. The parameter α can be known from multiple attempts.

The utilization of physical machines is calculated as follow:

$$Utilization_{PM} = \frac{t \cdot \sum NumberOfCores_i \cdot f_i \cdot Util_i + \beta \cdot SLAs}{t. \sum NumberOfCores_{PM} \cdot f_{PM}} \tag{2}$$

Basically, the nominator is the available capability of the physical machine and the numerator refers to the utilization of corresponding VMs plus the violation of SLAs multiplied by a parameter β which can be learnt.

They chose the 1998 French World Cup workload traces as the target data set to evaluate the performance, because that data set contains bursty and fluctuation of requests throughout the world, which is exactly the challenge they aimed to tackle.As for the evaluation of the algorithm, the authors took the violation of SLAs(Service Level Agreements) and the he VMs' demands and PM's supply into consideration. In their evaluation experiment, they compared the DEARS with other four resource scheduling methods, the results demonstrated that DEARS achieved better resource utilization and reduced the number of resource allocation violations compared to traditional resource scheduling approaches.

As for the strengths and shortcomings, we found that this paper focused mainly on automatic resource management and the elasticity of the scheduling, which improved efficiency, reduced human effort, and response times in handling workload fluctuations. On the other hand, the scalability of the proposed framework is not extensively discussed in the paper. And they did not compare the framework with state-of-the-art frameworks, and we saw no indication of the description of how to integrate this framework into existing cloud platforms.

3.2 A deep learning model based on a diffusion convolutional recurrent neural network

Mahfoudh Saeed Al-Asaly et al. proposed another DL-based model to forecast the workload for cloud resources. The core idea of the DCRNN model is to combine Graph Convolutional Network (GCN) with Recurrent Neural Network (RNN). It captures spatial correlation through graph convolutional operations and models temporal correlation using a recurrent neural network model. In DCRNN, graph convolution operations are used to learn spatial dependencies between nodes for information

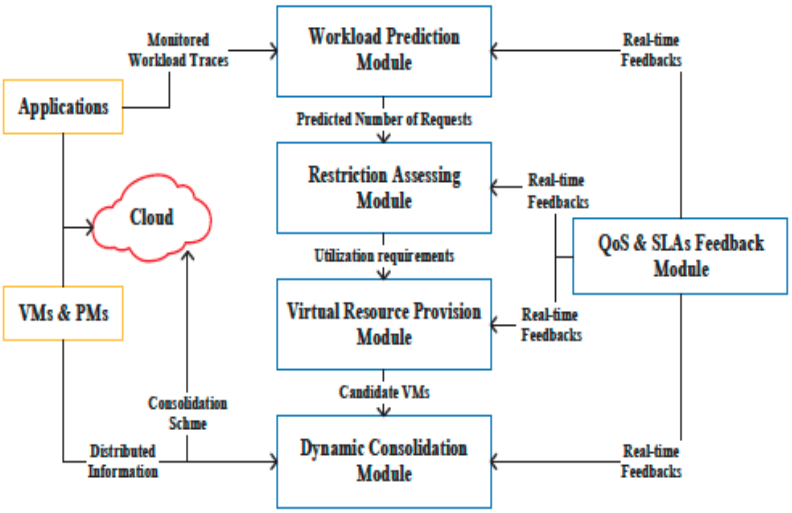


Figure 3: The framwork of DEARS

propagation in spatio-temporal data. These nodes can represent locations or regions in a transportation network in a city. By performing the convolution operation on the graph, the model can obtain information about the neighbours between the nodes and combine this information into the feature representation. [7]

The given data CPU utilization set consists of a series of timestamp. The DCRNN they applied was made up of two models: one encoding module and one decoding module. The architecture consists of two layers, each including four diffusion convolutional gated recurrent units. The details of the proposed structure can be seen in Figure 4 .

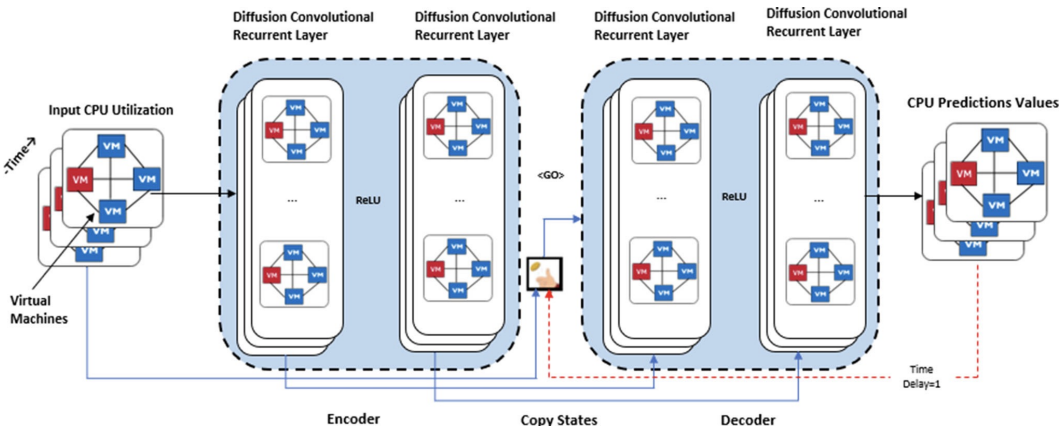


Figure 4: DCRNN system architecture

For model evaluation, they collected data from PlannetLab containing workloads for 10 different days and for each day with 288 samples. They then carried out data normalization to [0,1] scale and applied RMSE and MAPE as evaluation metrics. They compared the proposed model with other three models[4], conducting the experiment on 20 VMs that ran continuously for 10 days and on all VMs in PlanetLab. [12] The results showed that the proposed model outperformed other existing deep-learning approaches.

Regarding the strengths and weaknesses, this algorithm can learn directly from data without feature engineering, which can capture complex patterns in the data set. As for the evaluation, they provided

a comprehensive evaluation of the proposed DCRNN model using real-world traffic datasets and compared the proposed algorithms against several baseline methods to illustrate its superiority. However, we found that this paper fell short of implementation details such as how to configure the hyperparameters, moreover, they did not cover an analysis of its performance in real-world deployment scenarios.

4 Application of RL algorithm in Cloud Resource Scheduling

4.1 QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment

A cloud computing environment contains different tasks from different users. These tasks are executed by different cloud computing nodes, such as virtual machines. The dependency relationship between tasks varies and changes, which is hard for heuristic algorithms to handle.

The RL provides a method for agents to interact with a changing environment. The agent takes action from an action space and gets rewards from the environment. The state is also changed in certain state spaces. The rewards can be divided into immediate feedback and long-term feedback [19]. The agent learns strategy through some exploration-exploitation algorithms, such as the ϵ-greedy algorithm.

Tong Z, Deng X, Chen H et al propose a QL-HEFT reinforcement learning algorithm to schedule tasks efficiently. The QL-HEFT algorithm combines a heuristic scheduling method, HEFT, and Q-learning. It can be seen as an improved algorithm of HEFT to reduce the makespan. [15]

The model in which the algorithm is applied is the DAG-based task scheduling model. The basic model is shown in Fig 5. The users submit their jobs (tasks) to the cloud computing system. These tasks may be dependent on other tasks. Therefore, a directed acyclic graph (DAG) is used to describe the dependencies of tasks. The simple application of DAG is shown in Fig 5. The node represents a task with execution cost. The edge represents communication between two processors with a cost.

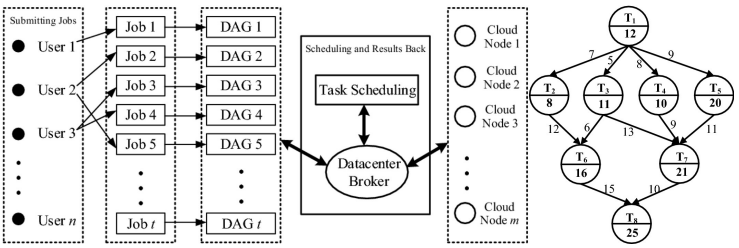


Figure 5: DAG model with a simple application

The goal of task scheduling is to minimise the execution time of tasks. It is achieved by the Q-learning algorithm, which is defined by the following formula.

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \left(r + \gamma \max_{a'} Q_t\left(s', a'\right) - Q_t(s,a) \right)$$

The s is the state that represents the task. The a is the action that assigns the task to a processor. The value $Q(s,a)$ is the cumulative value, which can be calculated iteratively. The r is the immediate reward which is *rank_u* [16]. The α is the learning rate and the γ is the discount factor. After getting the optimal task execution order, the algorithm allocates the task to an appropriate processor by the HEFT algorithm.

The algorithm is tested on CloudSim [3] using serval metrics such as makespan, speed up and CCR. When tested with different numbers of VMs and tasks, the QL-HEFT algorithm outperforms four similar algorithms. Another experiment shows that it accelerates the execution time of tasks from communication-intensive applications by 66.7%.

In this use case, the model combines heuristic algorithm and reinforcement learning and gets the optimal strategy to schedule tasks with dependencies. The results show the good performance of the model. But the computational complexity is a crucial concern. The large Q-table costs too much time and computational resources during updating.

4.2 Cloud Resource Scheduling With Deep Reinforcement Learning and Imitation Learning

As introduced in QL-HEFT, the RL finds an optimal strategy in a certain state and action space. For cloud resource scheduling, the state space might be too huge to calculate. Thus, deep learning is employed as an approximation function to avoid this situation. The new algorithm is called deep reinforcement learning, DRL.

Guo W, Tian W and Ye Y et al propose a deep reinforcement learning model called DeepRM_Plus[6]. The model is an upgraded version of DeepRM[8] through imitation learning. The model is not designed for dependent tasks but for independent tasks in online scheduling. In online scheduling, a task's arrival time and other critical information are usually unknown and may change before the task is executed. The online scheduling algorithm must select the appropriate task to execute based on the currently available resources and the attributes of the task.

Both two models work at a higher level of abstraction, combining cloud resources from different servers to schedule jobs. The following figure shows the resource state space(see Fig. 6). The job/task is decoded as a square in an image. The size of a square is the amount of resources required and the time slot.

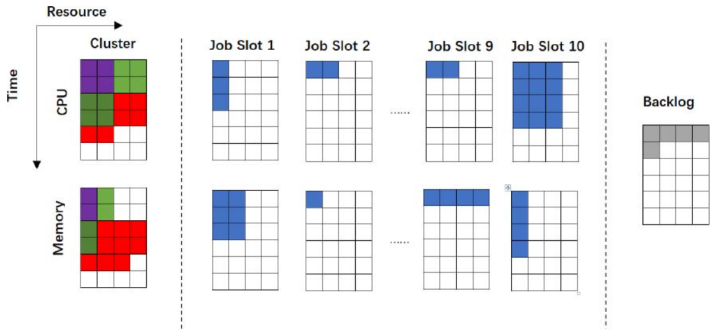


Figure 6: Resource state space

The state space image is fed into a six-layer CNN. The last fully connected layer outputs the action to allocate jobs. The agent will take action a and receive a reward r. The rewards are set for different objectives, such as minimising the average weighted turnaround time or minimising the average cycling time. The CNN is pre-trained by imitation learning of expert strategies. The model is shown below in Figure 7.

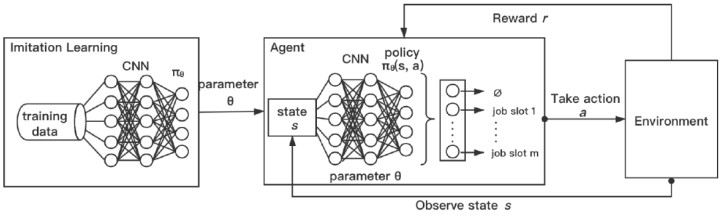


Figure 7: DeepRM_Plus model

In order to evaluate the model, the authors performed several experiments in terms of training, turnaround time and cycling time. The results show that DeepRM_Plus performs better than the SOTA model DeepRM.

As for this model, it shows the big potential of imitation learning which saves training time and cost. The DRL covers the shortage of exclusively huge state space. The model can handle different goals in resource scheduling. However, the way of representing tasks is more suitable for independent tasks.

5 Discussion and Conclusion

The main goals of cloud resource scheduling are to determine or predict the right amount of resources required for a task and to achieve optimal scheduling by using an efficient resource provisioning strategy. Another objective is to meet the requirements of QoS and SLA through parameters like cost, time, resource availability, and utilization by assigning suitable resources to jobs.[1]

In summary, machine learning algorithms can effectively handle resource management tasks in cloud resource scheduling. ML has been applied in various areas such as workload prediction, auto-scaling, VM arrangement, and task scheduling. Different machine learning techniques are suitable for different objectives in cloud resource scheduling. Prediction tasks can benefit from these techniques, such as SVM, neural networks, and deep neural networks. However, reinforcement learning and deep reinforcement learning are more suitable for finding optimal scheduling strategies. Overall, these ML techniques have contributed to improved performance metrics in cloud computing.

Machine learning algorithms excel in capturing target features during the scheduling process by learning nonlinear relationships and adapting to diverse behaviours. Reinforcement learning, in particular, demonstrates excellent performance by interacting with the environment and leveraging current information.

Looking ahead, deep learning and reinforcement learning continue to be prominent areas of research. Deep reinforcement learning has also shown significant potential in task scheduling. However, the complexity of these models introduces computational challenges. Addressing the issue of reducing computational complexity is a major consideration when applying ML techniques to practical cloud resource scheduling. Additionally, model interpretability and worst-case performance in different scenarios should be carefully considered. Furthermore, further research is necessary to explore the application of ML in complex and stochastic cloud environments.

References

[1] Aron, Rajni, and Ajith Abraham. "Resource scheduling methods for cloud computing environment: The role of meta-heuristics and artificial intelligence." Engineering Applications of Artificial Intelligence 116 (2022): 105345.

[2] Byun, Hyeran, and Seong-Whan Lee. “Applications of Support Vector Machines for Pattern Recognition: A Survey.” Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2388, Springer Berlin Heidelberg, 2002, pp. 213–36, doi:10.1007/3-540-45665-1_17.

[3] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software: Practice and experience, 2011, 41(1): 23-50.

[4] Deng, Li, and Dong Yu. "Deep learning: methods and applications." Foundations and trends® in signal processing 7.3–4 (2014): 197-387.

[5] François-Lavet V, Henderson P, Islam R, et al. An introduction to deep reinforcement learning[J]. Foundations and Trends® in Machine Learning, 2018, 11(3-4): 219-354.

[6] Guo W, Tian W, Ye Y, et al. Cloud resource scheduling with deep reinforcement learning and imitation learning[J]. IEEE Internet of Things Journal, 2020, 8(5): 3576-3586.

[7] Li Y, Yu R, Shahabi C, et al. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting[J]. arXiv preprint arXiv:1707.01926, 2017.

[8] Mao H, Alizadeh M, Menache I, et al. Resource management with deep reinforcement learning[C]//Proceedings of the 15th ACM workshop on hot topics in networks. 2016: 50-56.

[9] Moreno-Vozmediano, Rafael, et al. “Efficient Resource Provisioning for Elastic Cloud Services Based on Machine Learning Techniques.” Journal of Cloud Computing: Advances, Systems and Applications, vol. 8, no. 1, 2019, pp. 1–18, https://doi.org/10.1186/s13677-019-0128-9.

[10] M. Hassan, H. Chen and Y. Liu, "DEARS: A Deep Learning Based Elastic and Automatic Resource Scheduling Framework for Cloud Applications," 2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), Melbourne, VIC, Australia, 2018, pp. 541-548, doi: 10.1109/BDCloud.2018.00086.

[11] P. Ongsulee, "Artificial intelligence, machine learning and deep learning," 2017 15th International Conference on ICT and Knowledge Engineering (ICTKE), Bangkok, Thailand, 2017, pp. 1-6, doi: 10.1109/ICTKE.2017.8259629.

[12] Qiu, Feng, Bin Zhang, and Jun Guo. "A deep learning approach for VM workload prediction in the cloud." 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). IEEE, 2016.

[13] Qu, Chenhao, et al. "Auto-Scaling Web Applications in Clouds: A Taxonomy and Survey." ACM Computing Surveys, vol. 51, no. 4, 2018, pp. 1–33, https://doi.org/10.1145/3148149.

[14] Sniezynski, Bartlomiej, et al. "VM Reservation Plan Adaptation Using Machine Learning in Cloud Computing." Journal of Grid Computing, vol. 17, no. 4, 2019, pp. 797–812, https://doi.org/10.1007/s10723-019-09487-x.

[15] Tong Z, Deng X, Chen H, et al. QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment[J]. Neural Computing and Applications, 2020, 32: 5553-5570.

[16] Topcuoglu H, Hariri S, Wu M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE transactions on parallel and distributed systems, 2002, 13(3): 260-274.

[17] Yadav, Mahendra Pratap, et al. "Resource Provisioning Through Machine Learning in Cloud Services." Arabian Journal for Science and Engineering (2011), vol. 47, no. 2, 2022, pp. 1483–505, https://doi.org/10.1007/s13369-021-05864-5.

[18] Zhang, Xinqian, et al. "Energy-Aware Virtual Machine Allocation for Cloud with Resource Reservation." The Journal of Systems and Software, vol. 147, 2019, pp. 147–61, https://doi.org/10.1016/j.jss.2018.09.084.

[19] Zhou G, Tian W, Buyya R. Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions[J]. arXiv preprint arXiv:2105.04086, 2021.

[20] Zhu W, Zhuang Y, Zhang L. A three-dimensional virtual resource scheduling method for energy saving in cloud computing[J]. Future Generation Computer Systems, 2017, 69: 66-74.

Evolution of Data Centers and Emergence of Hyper-Converged Infrastructure

Anusha Ali Jeewon Heo Priyanka Roy

June 3, 2023

Abstract

Data centers emerged as a response to the increasing demand for efficient and scalable data storage and computing. They are constantly evolving to meet the ever-increasing volume of data and other requirements that follows from it. This paper explores the evolution of data centers, from their traditional forms to hyper-converged infrastructures. We introduce the technological milestones in data center architecture including software-defined data centers and newly emerging converged and hyper-converged infrastructures (HCI). We also delve into the emergence of HCI, the selection of HCI solutions, as well as its benefits and use cases. This article offers insights into the evolution of data centers leading up to HCI, thereby providing assistance in understanding various data center architectures and benefits of adopting HCI in businesses.

1 Introduction

The exponential growth of data in the rapidly growing IT industry has posed significant challenges and limitations for traditional data center infrastructures. As a result, data centers are undergoing a transformative evolution, shifting their focus from traditional architecture to converged and hyper-converged infrastructures (HCI). The primary objective of this transformation is to ensure high data availability in the face of ever-increasing data volumes [21]. A traditional data center design typically involves separate storage silos with different computing and networking-related components. This architecture often gives rise to challenges like more complexity, performance bottlenecks, limited scalability, and high costs. On the other hand, hyperconverged architecture offers a software-based integrated and consolidated approach to combine storage, computing, and networking resources into a single system. With HCI, all components are bound together [16]. In this paper, we aim to explain the evolution of data centers leading up to the hyperconverged infrastructure.

In Section 2, we introduce the types of data centers and traditional architectures, as well as their benefits and limitations. Section 3 illustrates the virtualization of data centers and the changes in architectural principles. Section 4 describes the concept of the software-defined data center. Section 5 explains convergence and hyper-convergence infrastructures in data centers. Section 6 concludes the paper.

2 Data Center

A data center is a facility that houses computer systems and their associated components, which provides storage capabilities and computing power [12]. Data centers emerged due to the rising need for organizations to manage large amounts of data, as well as the need for reliable operations.

2.1 Types of Data Centers

There are three major types of data centers which are discussed below

2.1.1 Enterprise Data Centers:

An enterprise data center is completely owned by a company dedicated to supporting a company's internal data processing needs and hosting critical applications. Many companies opt to have their data centers for higher control over their data, security, and regulations, such as GDPR[1] or HIPAA[2] [12].

2.1.2 Cloud Data Center

Cloud data centers, on the other hand, are hosted off-premise and serve their infrastructures to multiple users. The three major cloud service providers are Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. Cloud platforms now provide general services such as infrastructure as a service (IaaS), software as a service (SaaS), and platform as a service (PaaS) [14]. Other types of data centers include colocation, hybrid, edge, and micro data centers.

2.1.3 Colocation Data Centers

A colocation data center, often referred to as a "colo," is a facility that provides businesses the opportunity to lease dedicated space for their data center components. In essence, colocation data centers provide a highly secure and robust environment where companies can house their servers and other equipment. Colocation data centers are already equipped with backup generators, fire suppression systems, cooling systems, uninterruptible power supplies (UPS), and physical security measures such as security cameras and cooling systems. Internet services and connections to private networks are also often included in the facility.

2.2 Infrastructure Components

Servers, storage, and networking are considered as the building blocks of the data center infrastructure. Power supply and cable management, redundancy/disaster recovery (tier 1234), and environmental controls are also some other essential components of data center.

2.2.1 Servers

Servers are the most critical component of data center infrastructure. These can be considered as computers with high computing power to perform computationally expensive tasks. Data center servers can further be categorized as rack-mount servers, blade servers or mainframes.

2.2.2 Storage

The majority of the servers include direct attached storage (DAS), which is a local storage capability that keeps frequently used data close to the CPU. It can be considered as a cache of the servers. There are two other storages: storage area network (SAN) and network-attached storage (NAS). NAS devices usually have multiple storage devices like hard disks and solid-state drives. SAN also provides shared storage to the server but has a more complex infrastructure of storage management software, application servers, and storage servers.

2.2.3 Networking

Different types of routers, switches, and fiber optic cables are involved in the networking of data centers. These components aid in transferring the network traffic across different servers and clients.

[1] General Data Protection Regulation, `https://gdpr.eu`

[2] Health Insurance Portability and Accountability Act, `https://www.hhs.gov/hipaa/index.html`

2.3 Challenges

Traditional data centers have served as the backbone of computing infrastructure for many years. However, as technology advances and data volumes explode, traditional data centers face several limitations. We describe some of the challenges below.

2.3.1 Scalability

Traditional data centers often face problems in scaling effectively to incorporate the rapidly increasing data demands of modern applications. These infrastructures have a siloed architecture with different components for storage, computing, and networking. To meet the sudden increase in data, scaling each component independently requires complex integration and can result in inefficient resource allocation and underutilization. Additionally, traditional data centers are not able to scale dynamically to cater to unpredictable workloads.

2.3.2 Performance Bottlenecks

The architecture of traditional data centers can lead to performance bottlenecks. The interaction between separate components over complex network configurations introduces latency issues. Thus, impacting the responsiveness and overall performance of applications.

2.3.3 Cost and Infrastructure Complexity

Significant upfront capital investments are required to build the infrastructure of traditional data centers. Management of separate hardware components like storage devices, servers, and network equipment is required in traditional data centers. The complexity of integrating these components and the need for specialized expertise in each area contribute to higher costs and resource-intensive maintenance.

2.3.4 Management and Operational Challenges

Dedicated teams with expertise in different components of the whole infrastructure are often required in traditional data centers. This often increases operational complexity, response times, and inefficiencies. Ensuring high availability, and security management among the communication between different components can also become complicated to handle.

2.3.5 Limited Resilience and Disaster Recovery

Proper data replication strategies are hard to implement in the traditional data center infrastructure. Managing all the infrastructure configurations for different components gives rise to a lot of unknown challenges. Without implementing proper disaster recovery strategies, the risk of losing important data and high downtimes becomes hard to ignore in traditional data centers.

3 Virtualization

Traditionally, data centers ran on their own dedicated physical hardware resources. Nowadays, most data centers have virtualized their infrastructures, meaning they are not limited to the capabilities of the physical resources. Virtualization is the process of operating multiple virtualization machines (VMs) on a single physical server. This process is handled by a hypervisor, a software that allocates and manages resources between VMs. This type of virtualization is called **server virtualization** (or compute virtualization). With server virtualization, data centers can run from 10 to 40 virtual servers on a single physical server [7, Chapter 20]. Figure 1 shows the diagram of a network architecture of virtualized data center servers. Users of virtualized data centers must have access to their own set of VMs – or VDC (virtual data center).

In the following subsections, we lay out the benefits of virtualization and describe other forms of virtualization as well as the transformations of data centers that were made possible with the help of virtualization.

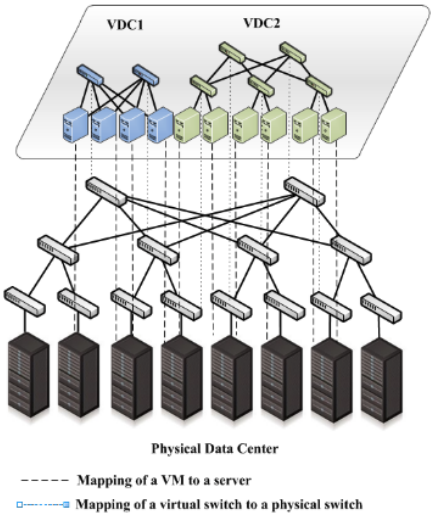


Figure 1: Diagram of virtualized data center network.

3.1 Benefits

Virtualization offers **flexibility** to data centers. With the physical infrastructures, deployment of servers used to take days or even months [7]. This has been significantly reduced to minutes thanks to virtualization. Moreover, it is capable of supporting legacy software or various operating systems. Furthermore, unlike physical servers, virtual data centers are inexpensive and simple to set up. This allows for increased **scalability** to quickly configure and provision infrastructures based on the need. In addition, virtualization **reduces the costs** of using data centers. Virtualized data centers are usually offered and maintained by third-party providers. This eliminates the maintenance cost of physical resources. Also, the providers usually have a consumption-based business model, meaning that they charge for the resources the companies actually use.

3.2 Software Defined Networking

Bari *et. al.* [3] states that the main challenge of virtualization arises from the network. We briefly describe each challenge as follows:

1. Multitenancy: managing shared data centers among multiple users.
2. Topology: the topology largely impacts the networking and the bandwidth between VMs.
3. Workload Mobility: live migration has strict network restrictions.
4. I/O Blocking: VMs on the same server share one network interface card (NIC). A hypervisor must carefully manage the traffic.

One of the main management issues is multitenancy, which is illustrated in Figure 2 [7, Chapter 20]. Each tenant requires isolated networking functions and different security and privacy levels. Most importantly, one tenant's traffic must be protected from another tenant.

Many of these networking challenges could be solved using software-defined networking, or SDN. SDN uses software-based controllers or application programming interfaces (APIs) to

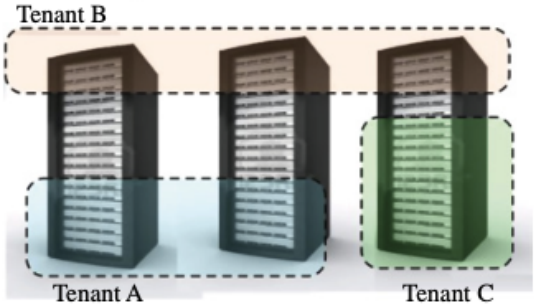


Figure 2: Multitenancy in data centers.

manage and configure network infrastructures [10, 22]. In traditional settings, the control pane, which is in charge of network devices, is bound with the data plane, which forwards network traffic. SDN, however, abstracts the control plane and decouples it from the data plane. Now, SDN can create and control virtual networks via software [26]. In essence, SDN can be defined with the following three principles: (1) separation of the control plane from the data plane, (2) centralized control, and (3) interface to communicate control of the networks [13].

There are three main components to a typical SDN architecture [20, 13]:

1. **Controllers** are the centralized entity in charge of managing network devices, translating requirements from applications, enforcing network policies, and monitoring network status.

2. **Applications** communicate network requirements to SDN controllers via a northbound interface (NBI).

3. **Networking devices** receive information from the controllers via a southbound interface (SBI) about moving data

SDN provides several advantages compared to hardware-based traditional networking. As the controllers are software-based, SDN provides more **flexibility** to control, **customize**, and provision network resources from a centralized interface, bypassing hardware entirely. Moreover, the abstraction of the controllers provides a better view of **security threats** [24]. SDN also has a positive effect on **reducing the costs** and complexity [9].

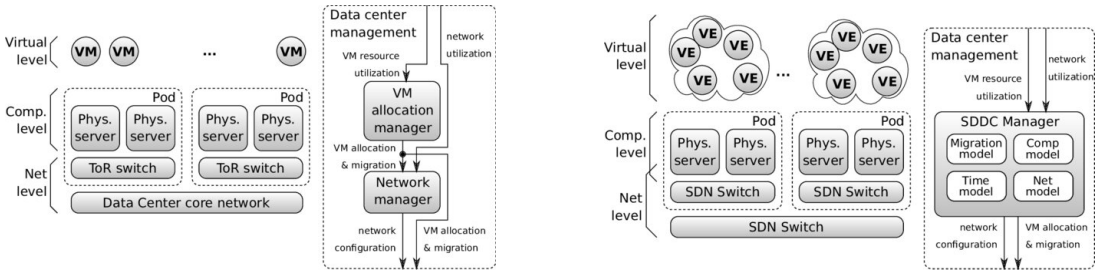
3.3 Infrastructure as a Service

Virtualization has enabled the infrastructure as a service (IaaS) on the cloud. IaaS offers businesses computing, storing, and networking resources on demand [27]. It allows the users to access their individual infrastructure which lie on the same physical hardware. The biggest benefit of IaaS is that it allows businesses to scale and shrink resources on a need basis.

4 Software-Defined Data Center

Software-defined data centers (SDDC) virtualize all infrastructure components in a data center including server, storage, and network with abstraction, pooling, and automation by software [11]. We have covered **server virtualization** in Section 3 and **network virtualization** (software-defined network) in Section 3.2. Storage virtualization among the virtualization of other components is newly introduced in SDDC. Like server virtualization, **storage virtualization** pools resources. More concretely, it gathers all blocks of storage in to a centralized shared pool from which they can be assigned to any VM on network as needed. This way, storage virtualization can improve flexibility and scalability [11].

SDDC is the integration of the entire infrastrcture components, creating a single centralized entity that manages the data center [6]. Figure 3 displays two diagrams, one for traditional and



(a) Traditional architecture [4].

(b) SDDC architecture [4].

Figure 3: Diagrams of data center architectures.

another for SDDC architecture. Figure 3b shows that a *SDN Switch* is added compared to the traditional model in Figure 3a. Canali *et. al.* use includes different "models" or components in the diagram, namely migration and time. A migration model is a component in charge of a migration process that occurs when a VE (virtual element – virtual machine or virtual element) is moved from one host to another one. A time model is there to reduce energy consumption or triggered when physical hosts overload [4]. While the terminologies or the components included in the diagram differ, the concept that SDDC provides an integrated process that combines all components remain the same. We can see that all requests are handled by an SDDC manager internally (Figure 3b).

4.1 Benefits

SDDC inherits the advantages of virtualization – flexibility and scalability. It can quickly provision resources and eliminates the time to set up new physical infrastructures. It also improves the performance with respect to compute, storage, and networking, by optimizing each components without making physical changes [11]. Also, SDDC allows policy-driven automation of provisioning and management, which "speeds delivery of resources and enhances efficiency" [11]. Moreover, SDDC is cost-effective in the long run. Resource pooling increases utilization, meaning that less fraction of infrastructure is idle. This leads to less purchase for physical resources, reduces real estate cost and power consumption.

4.2 Challenges

Switching to SDDC requires careful planning and management. First, standardization across teams is crucial. Aligning all teams, from procurement, development, and system administrator, can be time-consuming [11]. Furthermore, changing the data center architecture could cause application downtimes. Hence, it is important to plan ahead and proceed the adaption in phases to minimize unwanted downtimes [11]. However, once companies commit to this change, SDDC will improve the overall data center performances.

5 Hyper-Converged Infrastructure

Data centers have gone through different stages of evolution, starting from large mainframes to centralized storage servers and eventually to converged and hyper-converged infrastructures. Each stage aimed to address the shortcomings of the previous generation while bringing its own advantages and disadvantages [23]. In this section, we draw our attention to what Hyper-converged Infrastructure (HCI). Before that, it is important to understand its predecessor, Converged Infrastructure (CI), and how it leads to the emergence of HCI.

5.1 Path to Hyper-Converged Infrastructure

Definition Converged Infrastructure (CI) is referred to as a hardware-based approach to converging storage and processes that reduces compatibility issues, complex deployments, and overall costs. It works by using building blocks [2]. In other words, it is an ensemble of hardware and software components that, from a logical perspective, integrates all of these tasks into a single physical node. It is often packaged by a single vendor [25]. This strategy offers quicker deployment at times, and easier management. With CI one or more manufacturers define a validated design that has been pre-tested to operate consistently and reliably. This reduces the time to build the system.

HCI is a way to integrate multiple IT technologies, such as servers, storage, networking equipment, virtualization, and/or software applications into a larger solution. They allow the stacking of additional nodes to increase computing power, storage, and memory [23, 15].

Limitations of CI The drawback of this infrastructure is that it leads to vendor lock-in, which can result in fewer functionalities and limited options for customization. Also, adding new components to an already existing CI is a complex and expensive process [1]. CI solutions are largely adopted by larger enterprises moving from a traditional three-tier data center architecture [23]. Even though it requires specialized staff to manage the infrastructure it scales out horizontally very well. Figure 4 shows the constituents of a rack in CI [23].

Difference between CI and HCI Converged and hyper-converged infrastructures have similar objectives but have different implementations. The major distinction between the two solutions is that HCI is a software-defined solution and is made of software building blocks rather than physical hardware [2, 23]. While the network is still physically distinct and has a separate management plane, compute and storage are merged into a single solution and have a single control plane in HCI solutions [19]. A single, centralized management software may be used to manage all of the integrated technologies in a HCI while also allowing for the expansion of the basic systems with extra nodes to offer new capabilities. In short, HCI is an alternative to CI, it is software-defined. In contrast, CI is hardware-defined. Figure 5 shows a sample hyper-converged cluster [23].

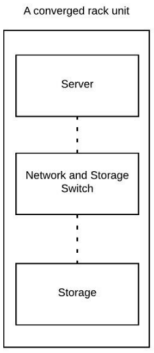


Figure 4: CI architecture.

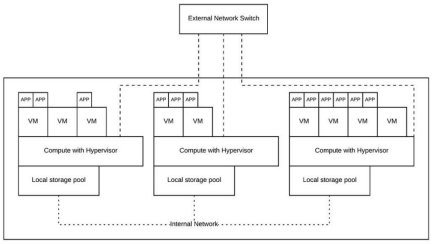


Figure 5: Three-node hyper-converged cluster

Overview of non-converged, converged, and hyper-converged infrastructures Figure 6 shows three diagrams that depict how the architecture has evolved from non-converged to converged and presently hyper-converged [18]. The leftmost figure shows how all the components are isolated in a non-converged infrastructure whereas the right-most figure depicts how the storage and server components are tightly coupled in contrast to the network layer, which is still isolated. This is a compact version of the middle figure that depicts a converged

infrastructure.

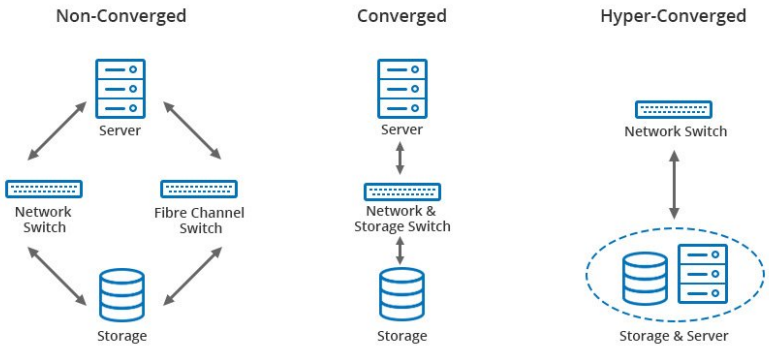


Figure 6: Architectural components of three types of Infrastructure [1].

5.2 Benefits of Hyper-Converged Infrastructure

- **Balance-workload** The vendor lock-in problem with CI solutions is addressed by HCI, which is built on common hardware that is easily replaceable. Bein a software-defined storage framework, each node's storage controller essentially is a service that runs continuously [23]. There is no downtime or extra hardware needed when HCI components are temporarily moved to different nodes for repair. Users can continue working normally throughout the maintenance window because it is transparent to them [18]

- **Cost -Effective** Numerous small to mid-sized businesses are drawn to the utilization due to the flexibility of adding hardware from any vendor and the ease of management along with considerably faster implementation (in hours). The price of the underlying hypervisor affects how much HCI costs. For instance, the cost of commercial versus open-source systems differs. Interoperability and complexity issues are also reduced or resolved as a result of lower operational costs (and TCO).

- **Effective resource distribution and data utilization** VMs can be moved between hosts. This makes it possible to reuse VM-specific functionality across several platforms. An organization can use HCI to give teams access to specialized environments, such as those for testing, development, or applications that are exclusive to a given industry [18]. Storage utilization is increased by keeping storage close to the computation and dividing access to it throughout the group of compute servers, which also reduces storage IO latency [23]. By simply adding a new node (computer and storage) to the current cluster (group of nodes), the infrastructure may be scaled out linearly [23].

- **Easy Management** HCI platform is significantly simpler to manage, operate, and monitor than traditional infrastructure. IT administrators may set up the HCI system to automatically allocate resources, conduct duplicate activities, eliminate all administrative silos, and execute data backup and restore as normal because it is software-driven [18]. In a variety of virtual environments, HCI provides single-view analysis. This aids management to perform high-level system evaluations and monitoring inside a company. In order to track the effectiveness of virtual environments, modern HCI adds AI components. Moving HCI workloads from one guest machine to another is simple. In agile contexts, this functionality facilitates the quick development and deployment of software [17]

Drawbacks If the Operating System license is per CPU core, this is a disadvantage to customers who only require storage as horizontal scaling in HCI is expensive because a completely new node has to be added even if they just want to scale up computation or storage capacity. HCI solutions have resource consumption overhead as a result of various management applications. [23].

5.3 Selecting the right HCI Solution

There are various HCI solutions available in the market. There are numerous factors that are important to consider when evaluating HCI solutions. Depending upon the use-case an organization might choose to go for a specific implementation of HCI. Nowadays organizations might consider some of the following characteristics when trying to choose the right HCI solution for them :

1. **Edge-core cloud integration** When it comes to linking current infrastructure, clouds, and edge services, organizations have a wide range of requirements. For instance, a company could merely use the cloud's storage tier. When switching cloud providers, it can also wish to replicate or convert configurations. An HCI solution should enable an organization to modify, update, and adapt as infrastructure requirements change [8]. Integration with various forms of storage, processing, and network resources as well as cloud services is essential for the HCI platform. If an organization wants to start container orchestration as a future endeavor, having container support may also be an essential requirement.

2. **Analytics** Since HCI aggregates storage, computation, and network resources, organizations must be able to monitor resource allocation, use, and health. It might be quite advantageous to select an HCI platform that provides centralized dashboards with premade and custom metrics as well as reporting capabilities [17]. It should enable means to drill deeper into data, acquire information on what is happening, and provide access via a single dashboard. Additionally, this aids in capacity planning and trend analysis [8].

3. **Storage management** An ideal HCI solution should let easy integration and configure a wide range of storage systems. Additionally, mapping these storage systems to the changing requirements of an organization's IT ecosystem should be feasible. Some modern HCI systems are offering support for NVMe-OF (non-volatile memory express over fabrics), a technology that simplifies storage re-architecture using flash memory.

5.4 Use-case scenarios for HCI solutions

HCI is cost-effective, it offers a low price point per unit. It is also extremely easy to roll out and is highly scalable. Following are the scenarios where HCI could be used.

1. **DTAP Environment** Before applications are deployed to production settings, HCI is frequently used to develop, test, and ensure the quality of the applications. The isolation of virtualized servers prevents problems from affecting other active applications [8, 2]. By replicating resources from the production environment, HCI offers a suitable match for development and testing environments. In the case of HCI, the development, testing, and fine-tuning phases of a test environment do not overlap with the production cluster [5]. In this manner, HCI offers a reliable and affordable test environment that does not interfere with the performance of current IT clusters. The processes are moved to the production side once the testing and development phases are over and they have stabilized.

2. **VDI and ROBO** Hosting VDI solutions is one of the key use cases for HCI. VDI is the practice of running desktop operating systems in the cloud using virtualized desktop software. Users of laptops or mobile devices may remotely access virtual desktops thanks to VDI. A virtual machine (VM), which is kept on a centralized server, houses the VDI

software. Examples of VDI use cases include remote workers connecting through VDI software to an organization's platform or support center employees connecting remotely to a main workstation to react to client inquiries around-the-clock [8, 2].

3. **Typical organizational candidates** HCI is a useful tool in scenarios where different applications are required to concurrently perform day-to-day processes, for example, finance, HR, and IT support [18]. Another example is where storage needs to be scalable on demand because there is a surge in online requests [18]. HCI can also be used where a reliable repository is needed for large amounts of structured and unstructured data that is in low demand such as a library archive [18]. A small start-up company that is starting to grow might not be open to the complexities of setting up a big data center which could bring high costs, and complexity in setting up or might demand expertise and knowledge. They might need an infrastructure that is cost-effective, scalable, simple to manage, and flexible.

Remark While the architecture of data centers have evolved, prior solutions are still relevant. The architecture that an organization adopts depends on its requirements. This means HCI might not be a solution for every organization. They must carefully investigate and check if HCI is the right option for them. After reading many scientific articles we also tried to find some use-case scenarios for HCI. This led us to realize that if it is a small enterprise or a start-up whose priority is to implement its solutions quickly and affordably then it might choose converged or hyper-converged infrastructure solutions over a typical three-tier design. But even large organizations can also opt for HCI because of its numerous benefits as mentioned above over a three-tier architecture. When an organization's main concern is data, servers with centralized storage are implemented to provide the data with high availability and security with centralized administration [23].

6 Conclusion

This paper has provided a comprehensive overview of the evolution of data centers, from mainframes to the newly emerging hyper-converged infrastructure (HCI). We defined the types and components of data centers, their traditional setups, and their revolution via virtualization technology. We also discussed the concepts of software-defined networks (SDN) and infrastructure as a service (IaaS), which stem from the application of virtualization. We then explained the software-defined data center (SDDC), which takes a step further in virtualizing data centers. Finally, the paper introduces converged infrastructure (CI) and HCI, which aim to converge data center components into a software-defined platform. We delve deep into the concept of HCI and its benefits, as well as the selection of HCI solutions and their use cases.

In conclusion, the evolution of data centers has been a dynamic process driven by technological advancements and the need for greater efficiency, scalability, and agility As data centers continue to evolve, HCI presents an exciting opportunity for organizations to revolutionize their data center strategies and unlock new levels of performance and innovation.

In conclusion, technical improvements and the demand for higher efficiency, scalability, and flexibility have pushed the dynamic growth of data centers. As data centers continue to evolve, HCI offers organizations an opportunity to transform their data center setups and achieve a higher level of performance.

References

[1] Azeem, S., and Sharma, S. Study of converged infrastructure & hyper converge infrastructure as future of data centre. *International Journal of Advanced Computer Research 8* (09 2019), 900.

[2] Azure, M. Azure Stack HCI: Use cases and scenarios. `https://azure.microsoft.com/mediahandler/files/resourcefiles/azure-stack-hci-use-cases-and-scenarios/MSFT-AzureStackHCIBoM-UseCases-WP-Final.pdf`, 2020. [Accessed 25-May-2023].

[3] Bari, M. F., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., and Zhani, M. F. Data center network virtualization: A survey. *IEEE Communications Surveys & Tutorials 15* (01 2013), 909–928.

[4] Canali, C., Lancellotti, R., and Shojafar, M. *An Optimization Model to Reduce Energy Consumption in Software-Defined Data Centers: 7th International Conference, CLOSER 2017, Porto, Portugal, April 24–26, 2017, Revised Selected Papers*. Springer New York, NY, 07 2018, pp. 137–156.

[5] DinCloud. Use Cases for Hyper Converged Infrastructure (HCI). `https://www.dincloud.com/blog/hyper-converged-infrastructure-hci-use-cases`. [Accessed 24-May-2023].

[6] Fung, H. P. A Glimpse into Software Defined Data Center UM GLIMPSE EM SOFTWARE DEFINED DATA CENTER. *Journal of Management of Roraima 4* (07 2014), 34–49.

[7] Geng, H., Ed. *Data Center Handbook*. John Wiley & Sons, Ltd, 2021.

[8] Greengard, S. Top 10 Hyperconverged Infrastructure (HCI) Solutions. `https://www.datamation.com/data-center/top-10-hyperconverged-infrastructure-hci-solutions/`, 2020. [Accessed 24-May-2023].

[9] Hampel, G., Steiner, M., and Bu, T. Applying software-defined networking to the telecom domain. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2013), pp. 133–138.

[10] Horvath, R., Nedbal, D., and Stieninger, M. A literature review on challenges and effects of software defined networking. *Procedia Computer Science 64* (2015), 552–561. Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2015 October 7-9, 2015.

[11] IBM. Software-Defined Data Centers. `https://www.ibm.com/topics/software-defined-data-center`. [Accessed 22-May-2023].

[12] IBM. What is a data center? `https://www.ibm.com/topics/data-centers`, 2021. [Accessed 26-May-2023].

[13] Jammal, M., Singh, T., Shami, A., Asal, R., and Li, Y. Software defined networking: State of the art and research challenges. *Computer Networks 72* (2014), 74–98.

[14] Kant, K. Data center evolution: A tutorial on state of the art, issues, and challenges. *Computer Networks 53*, 17 (2009), 2939–2965. Virtualized Data Centers.

[15] Magsi, Z., Koondhar, M. Y., Hyder Depar, M., Pathan, Z. H., Memon, F.-U.-D., and Solangi, S. Conceptual framework transformation of converged infrastructure (ci) into hyper converged technology for virtualization of server infrastructure. In *2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS)* (2020), pp. 1–4.

[16] Miller, A. Converged vs Hyperconverged Infrastructure: The Differences Between CI & HCI. https://www.bmc.com/blogs/converged-infrastructure-vs-hyper-converged-infrastructure/#:~:text=Converged%20Infrastructure%20(CI)%20is%20a,to%20converging%20storage%20and%20processes., 2021. [Accessed 26-May-2023].

[17] Mudrakola, S. What Are the Best Hyperconverged Infrastructure (HCI) Solutions on the Market? https://petri.com/top-5-hyperconverged-infrastructure-platforms/, 2023. [Accessed 24-May-2023].

[18] Paessler. IT Explained: HCI. https://www.paessler.com/it-explained/hci. [Accessed 24-May-2023].

[19] Prabowo, R. J., Hidayanto, A. N., Sandhyaduhita, P. I., Azzahro, F., and Chairunnisa, A. The determinants of user's intention to adopt hyper-converged infrastructure technologies: An integrated approach. In *2018 International Conference on Information Technology Systems and Innovation (ICITSI)* (2018), pp. 306–311.

[20] Priyadarsini, M., and Bera, P. Software defined networking architecture, traffic management, security, and placement: A survey. *Computer Networks 192* (2021), 108047.

[21] Ramanathan, M., and Narayanan, K. Analysis of substantial growth on data center in virtualization local area network applications. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)* (2019), pp. 1097–1100.

[22] Sharma, R. A review on software defined networking. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* (03 2021), 11–14.

[23] Shetty, A. J., and Ganashree, K. Comprehensive review of datacenter architecture evolution. In *2020 International Research Journal of Engineering and Technology* (2020), vol. 7, pp. 7420–7426.

[24] Urrea, C., and Benítez, D. Software-defined networking solutions, architecture and controllers for the industrial internet of things: A review. *Sensors 21*, 19 (2021), 6585.

[25] Veiga, A. P. Hyper converged infrastructures: Beyond virtualization, 2017.

[26] VMWare. What is Software-Defined Networking (SDN)? https://www.vmware.com/topics/glossary/content/software-defined-networking.html. [Accessed 26-May-2023].

[27] Yang, C.-T., Huang, K.-L., Chu, W. C.-C., Leu, F.-Y., and Wang, S.-F. Implementation of cloud iaas for virtualization with live migration. In *Grid and Pervasive Computing* (Berlin, Heidelberg, 2013), J. J. J. H. Park, H. R. Arabnia, C. Kim, W. Shi, and J.-M. Gil, Eds., Springer Berlin Heidelberg, pp. 199–207.

How Can Peer-to-Peer Deep Learning Enhance Edge Computing in IoT Devices?

Rounak Vyas
Universiteit van Amsterdam
r.vyas@student.vu.nl

Kemal Sheker
Universiteit van Amsterdam
k.sheker@student.vu.nl

Punya Kapoor
Universiteit van Amsterdam
p.kapoor@student.vu.nl

Abstract

This literature study explores the potential of peer-to-peer deep learning to improve edge computing in IoT devices. In order to improve the functionality of IoT devices, the study examines a number of strategies, such as federated learning, peer-to-peer deep learning, and coupled orchestration of edge and fog computing. The study focuses on the challenges posed by resource constraints and the need for collaborative learning techniques to decrease these limitations. The integration of deep learning with edge computing in IoT is crucial for efficient data processing and real-time decision-making. The study also talks about the peer-to-peer security issues that come with IoT edge computing and possible solutions to these issues.

1 Introduction

The exponential growth of Internet of Things (IoT) devices has led to a massive influx of data at the network edge. These devices, ranging from smart sensors to wearables, generate a continuous stream of data that presents both opportunities and challenges for efficient processing and analysis. However, the limited computational resources and energy constraints of edge devices make it challenging to perform computationally intensive tasks like deep learning. To address these challenges, researchers have explored the integration of edge computing and peer-to-peer deep learning as a promising approach to enhance the capabilities of IoT devices.

Collaborative learning in edge computing has gained significant attention as it enables the utilization of distributed data sources and facilitates knowledge sharing among edge devices. Several research papers have contributed to this area of study. Chen and Ran [1] provided a comprehensive review of deep learning with edge computing, highlighting the potential benefits and challenges. They discussed various techniques and methodologies employed in the integration of deep learning with edge computing, shedding light on the advancements in this field.

Moreover, Šajina, Tankovic, and Ipsic [2] proposed a novel approach called peer-to-peer deep learning with non-IID data. Their work demonstrated the feasibility of leveraging peer-to-peer communication for deep learning tasks with non-identically distributed (non-IID) data in IoT networks. This approach enables collaborative learning and knowledge exchange among edge devices, allowing them to collectively train deep learning models without relying on centralized servers. By utilizing non-IID data, the proposed approach addresses the heterogeneity of data distributions across edge devices, leading to improved learning performance.

In addition to the works mentioned above, Nguyen et al. [3] conducted a comprehensive survey on federated learning for the Internet of Things (IoT). Their study explored the potential of federated learning in IoT scenarios, considering aspects such as data privacy, communication efficiency, and model accuracy. The survey provided insights into the methodologies, challenges, and future directions of federated learning in the context of IoT.

Universiteit van Amsterdam, Amsterdam, The Netherlands.

Motivated by the findings of Chen and Ran [1], Šajina et al. [2], and Nguyen et al. [3], this research paper aims to investigate the potential of peer-to-peer deep learning in enhancing edge computing for IoT devices. The objective is to explore the effective employment of collaborative learning techniques in an edge computing environment to mitigate the limitations of individual edge devices and improve overall performance.

In the subsequent sections, we will explore the potential of peer-to-peer deep learning in enhancing edge computing for IoT devices. We will discuss federated learning for IoT, peer-to-peer deep learning in IoT environments, collaborative deep learning in resource-constrained IoT edge devices, and the integration of deep learning with edge computing. By examining these topics, we aim to provide a comprehensive understanding of how peer-to-peer deep learning can improve the capabilities of IoT devices at the network edge.

2 Peer-to-Peer Deep Learning and Edge Computing

In this section, we will dive deep into how deep learning can change the edge computing scene. We will explain how P2P IoT, which has become widespread, can get better with deep learning and how deep learning is done. We will talk about the differences between IoT and edge computing devices from cloud computing, their resource constraints, and low-latency expectations. Next, we'll discuss the P2P security issues that come with IoT edge computing and possible solutions to these issues.

2.1 P2P Deep Learning in IoT Edge Computing

In recent years, integrating peer-to-peer (P2P) deep learning techniques with edge computing in IoT devices has garnered significant attention. This section explores the advancements, challenges, and potential research directions in this domain.

One area of interest is federated learning for IoT, which enables IoT devices to collaboratively train models without exchanging raw data. Nguyen et al. [3] provide a comprehensive survey of federated learning, discussing its recent advances and future directions. This approach addresses privacy concerns and reduces communication costs, making it suitable for IoT edge computing scenarios. However, challenges such as heterogeneous device capabilities, network connectivity, and data heterogeneity need to be addressed to fully exploit the potential of federated learning in IoT edge environments.

Another approach that has gained attention is peer-to-peer deep learning in IoT environments. Joshi et al. [4] delve into recent advances and research challenges in this field. P2P deep learning enables IoT devices to collectively train models through peer-to-peer communication, leveraging their computational capabilities. This approach offers reduced latency, improved scalability, and efficient resource utilization. However, it also poses challenges related to resource constraints and network dynamics. Optimizing communication protocols, load balancing, and fault tolerance mechanisms are important research directions in this context.

Collaborative deep learning in resource-constrained IoT edge devices is another important area of research. Zhang et al. [5] propose a trusted and collaborative framework for deep learning in IoT, focusing on efficient model aggregation and exploring the trade-off between model accuracy and resource consumption. Collaborative deep learning in such environments harnesses the collective intelligence of IoT devices while respecting their limitations. Addressing challenges related to limited computational power, energy constraints, and communication overheads is crucial for effective collaborative deep learning in IoT edge computing.

The integration of deep learning with edge computing in IoT is crucial for efficient data processing and real-time decision-making. Li et al. [6] discuss various approaches for incorporating deep learning models into edge devices, enhancing their capabilities to handle IoT-generated data. This integration offers benefits such as reduced latency and improved privacy. Their work emphasizes the significance of edge computing in enabling intelligent processing at the network edge. Overcoming challenges related to limited storage capacity, energy efficiency, and model deployment on resource-constrained edge devices is critical for the successful implementation of deep learning in IoT edge environments.

Furthermore, Šajina, Tankovic, and Ipsic [2] proposed peer-to-peer deep learning with non-IID data, which addresses the heterogeneity of data distributions across IoT devices. Their work focuses

on enabling collaborative learning and knowledge exchange among edge devices. By leveraging peer-to-peer communication, this approach allows edge devices to collectively train deep learning models, minimizing reliance on centralized servers and reducing latency and bandwidth consumption.

To enhance the understanding of P2P deep learning in IoT edge computing, including a figure illustrating the architecture or workflow can be beneficial. Figure 1 illustrates a possible architecture for P2P deep learning in IoT edge computing, showcasing the collaborative training process among edge devices. This figure can help visualize the concept and highlight the interactions between devices.

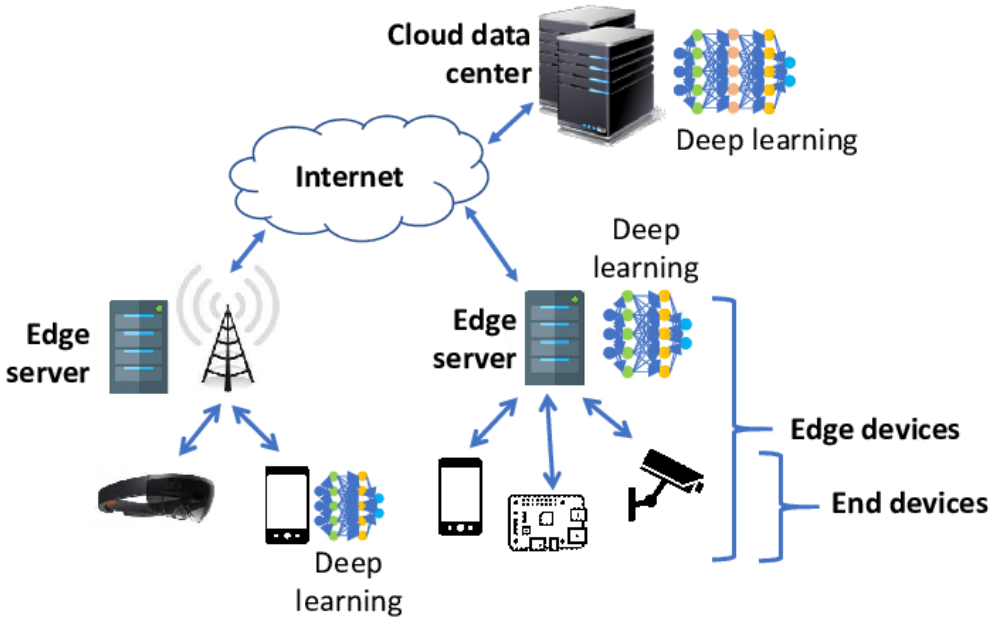


Figure 1: P2P Deep Learning Architecture in IoT Edge Computing[1]

Based on the insights from these studies, the research on peer-to-peer deep learning in IoT edge computing is poised to address challenges such as resource constraints, privacy preservation, and network dynamics. Future research directions may involve optimizing communication and computation efficiency, developing robust and secure algorithms, exploring innovative approaches for federated learning and collaborative deep learning, and investigating the integration of edge intelligence with cloud-based infrastructures. These advancements will pave the way for intelligent IoT systems with enhanced capabilities and efficient data processing at the network edge.

2.2 Resource-Constrained Environments and Low-Latency Requirements

In this section, we will talk about doing deep learning on IoT edge devices. These devices are limited by computational resources, restricted power budgets, and the need for near real-time decision-making. In addition to the benefits that deep learning provides to these devices, it addresses the difficulties of doing deep learning on these devices and the reasons for these difficulties. Possible solutions to these problems and evaluation of these solutions will be made.

Due to budget constraints, fail-over options are limited in IoT edge computing compared to traditional centralized cloud-based environments. Saurabh Bagchi et al.(2019)[7], listed various dependability issues on IoT edge computing. Based on this article, we can address the dependability problems we will encounter while doing deep learning in edge computing. One of the first problems is the lack of a common edge computing standard for IoT devices. The article suggests that new standards or mediation layers should be designed to coordinate devices and provide useful functionality, such that an edge device can seamlessly communicate with and control multiple end-user devices[7]. This is one of the problems in front of edge computing becoming more widespread and popular. Deep learning in centralized cloud computing is already an established and applied method. However, there are numerous reasons why existing cloud computing standards cannot be immediately transferred

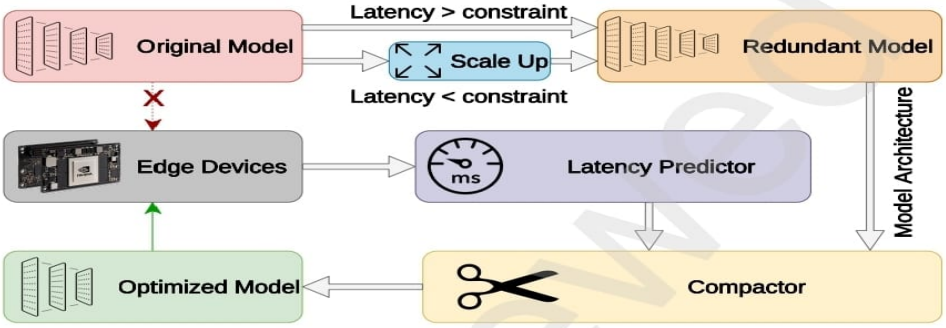


Figure 2: The overview of Latency-Constrained DNN Architecture Learning for Edge Systems.[10]

to edge computing. In contrast to edge computing, which involves more frequent and lighter communication, cloud computing typically involves heavier computational tasks and large amounts of data transfer. As a result, the latency and bandwidth problems associated with service delivery must be handled differently due to the resource-constrained environments of edge computing[7]. In this area, developers and manufacturers in the industry need to take steps together to build a framework for edge computing.

Another method to overcome the difficulty of deep learning in the resource-constraint environments of edge computing is the Joint orchestration of Edge and Fog computing[8]. The fog nodes, which are computing devices located at the edge of the network, are responsible for processing and analyzing the data in real-time. One opportunity in this orchestration is resource utilization enhancement introduced by L.Comardi et al.(2018)[8]. It offers the distribution of computing and networking tasks across both edge and fog resources, including any type of devices that possess networking and computing capabilities. This naturally creates a larger pool of resources dispersed close to the end consumers, enabling better computational gains and improved resource usage. In this article, deep learning is not mentioned, but in this way, an environment where deep learning algorithms can be run better in these systems is created. In this area, it is possible to investigate the use of deep learning further and understand what can be done in the future.

Another challenge of working in a resource-constrained environment is the lack of efficient tools to protect data privacy and security at the edge of the network. Shi, W., et al.(2016)[9], addresses this issue. Some devices have limited resources, making it difficult to implement existing security protection techniques on them because they are resource-hungry[9]. This means that security measures that require a lot of resources, such as encryption or complex authentication protocols, may not be feasible to implement on these devices without significantly impacting their performance and functionality. For example, using these security applications on an AR device that requires low latency will increase latency and cause slowdowns in a device that requires low latency. There are not many tools in this field for now.

There is research about Latency-Constrained DNN(Deep Neural Network) Architecture Learning for Edge Systems. Shuo Huai et al.(2023) propose a latency-oriented neural network learning method to optimize models for high accuracy while fulfilling the latency constraint in edge systems[10]. They introduce a universal hardware-customized latency predictor by using a Backpropagation in Neural Network. This procedure learns a model that satisfies the latency constraint by only a one-shot training process. One-shot training describes a DNN training in which a model is trained on a tiny dataset with one example per class or category. With this method, they can do deep learning faster using fewer resources, which is what we need in edge computing. Within the main framework, to prevent time-consuming on-device measurements during each Zero training phase, we provide a machine learning-based latency predictor that is integrated into our adaptive learning architecture[10]. Experiments were made in latency-constrained environments to see how well this framework was developed in the end. The results of their experiments showed that this framework has high accuracy in latency-constrained edge systems. We can foresee that this research and the framework created can be done for other constraints in edge computing. For example, explore the trade-off between energy

efficiency and latency in edge systems. By developing new approaches, a balance between energy and latency can be established and deep learning can be done.

2.3 Data Privacy and Security in P2P Deep Learning

Large amounts of data can be generated by Internet of Things (IoT) edge devices, much of it related to human behavior and activity. Individuals face serious privacy risks when personal data is collected and machine learning models are trained on a central cloud server. There are also difficulties when sending this data to the cloud. To create high-performance prediction models, insights based on machine learning, particularly deep learning, considerably benefit from enormous amounts of data. Moreover, there are significant cybersecurity problems due to the growing use of mobile devices. Deep learning (DL) models are used by mobile services and applications for the modeling, categorization, and identification of complex data, such as photos, audio, video, or text. Users gain from the numerous services and applications that these devices provide, but at a significant cost: the privacy of their data. Mobile services gather a wide variety of user information, including sensitive personal information, images, videos, clinical data, banking information, etc. Big firms use all of this data that has been gathered from consumers to train their worldwide DL models, which obviously raises severe privacy concerns. In this section, we will talk about how P2P deep learning helps with overcoming the above challenges. In addition to this, we will review several privacy-preserving deep learning methods that can enhance edge computing in IoT devices.

2.3.1 Privacy Preserving in Federated Learning

Briggs et al.[12] review privacy-preserving federated learning for IoT devices. Federated learning extends the idea of distributed machine learning, making use of decentralized learning. The decentralized distribution of training data does not rely on a centralized parameter server to coalesce model updates from edge devices but instead allows the edge devices to communicate with one another, resulting in each edge device performing aggregation on data from the parameters it receives. To simplify, rather than randomly partitioning a dataset to many compute nodes, model training occurs on edge devices using the distributed data owned by individual users. In the field of robotics, federated learning is used to secure privacy-sensitive tasks and more generally to help different robots share imitation learning methodologies. In edge computing environments, federated learning has been used to predict demand in the deployed applications and for improving content caching mechanisms.

Due to the decentralized nature of client participation required for federated learning, the protocol is vulnerable to adversarial attacks. Bagdasaryan et al.[13] outline techniques for poisoning the global model in a federated learning environment by using an adversary as a client. It builds an update that endures the averaging process and significantly affects or replaces the global model. Additionally, even though the training data itself is never exposed to a third party, it is possible that the parameter adjustments could reveal something about an individual's training data.

Bonawitz et al. [14] propose an approach to preserve the privacy of individual users in a federated learning setting. Their work uses secure multiparty computation (MPC) to compute averages of model parameter updates from individual edge devices in a secure manner. The approach uses secure aggregation that computes a multiparty sum where no party reveals its updates. The benefit of this approach is that edge devices can share an update knowing that the service provider will only see that update after it has been averaged with those of other edge devices. However, their work focused primarily on the setting of mobile devices, where dropouts are trivial and communication is expensive.

In Figure 3, user edge devices interact with cloud-hosted models in the cloud-centric approach to machine intelligence, creating logs that can be utilized as training examples. Numerous users' logs are merged and utilized to enhance the model, which is subsequently used to fulfill requests from new users. Federated learning sends ML models to consumers' devices where they are locally trained and assessed. The server receives summaries of updated models, which are then combined into a new model and distributed to user devices. When secure aggregation is introduced to federated learning, the cloud provider only learns the aggregated model update since the aggregation of model updates is logically performed by the virtual, incorruptible third party caused by the secure multiparty communication.

The time it takes for federated learning to learn a model relies on the number of training steps and the ML parameter transmission per step. This is because federated learning requires edge devices to

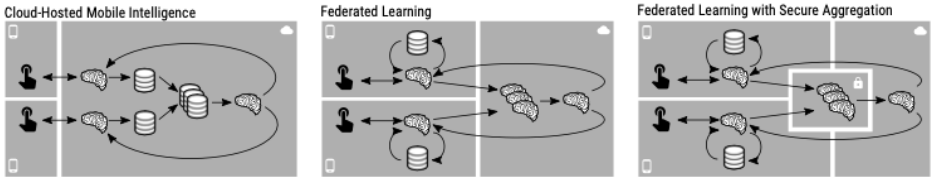


Figure 3: Comparision between cloud-hosted mobile intelligence, federated learning, and federated learning with secure aggregation.

share their ML parameters iteratively. In reality, several participating edge devices frequently transmit model parameters through communication networks with constrained resources, such as wireless networks with constrained bandwidth and power. As a result, the frequent transfer from edge devices results in a delay that may be orders of magnitude longer than the time required to train an ML model. Delay in communication is hence a significant bottleneck in federated learning. Chena et al. [15] propose an approach to improve the convergence time and the training loss using a communication-efficient federated learning framework. In this framework, a probabilistic device selection system is created such that edge devices that can improve the convergence speed and training loss have a greater probability to be selected for model transmission. Moreover, a quantization technique is suggested to decrease the volume of model parameters transferred among devices and an effective wireless resource allocation strategy is created in order to further shorten the convergence time.

2.3.2 Privacy Preserving in Fog Computing

Apart from federated learning, fog computing environments have been improved to include privacy preservation frameworks. Gutiérrez et al. [16] propose a similar framework that uses a distributed deep learning approach. Instead of disclosing their sensitivity to the cloud server, Internet of Things (IoT) end nodes communicate a portion of the users' data with a nearby Fog node that has been masked with Gaussian noise. The framework is designed to preserve users' privacy. It consists of a three-level architecture which consists of multiple edge devices at the bottom (also called V devices) or end nodes, Fog nodes in the middle level, and the cloud server at the top level. The edge devices do not reveal their data to the cloud server by only sharing a fragment of their data with a nearby Fog node. In addition to this, Gaussian noise has been introduced to provide an additional layer of privacy to users' sensitive data. The edge devices have all the training data to train the model. Each device opts for a specific portion of the dataset and preprocesses the data. Once the data is preprocessed, it is sent to its corresponding Fog node. In this fog-embedded architecture, the Fog nodes train the model in addition to preprocessing the data in the end nodes, leaving the cloud server with simply the duty of validating and updating the model so that it is accessible to other fog nodes.

2.3.3 Private P2P Network for Distributed Machine Learning

In 2021, researchers (Zhou et al.) introduced a network architecture focused on enhancing efficiency in communication, computation, and time. The architecture draws inspiration from Dempster-Shafer's theory and utilizes a Scalable Chord Peer-to-Peer Network. Additionally, it suggests a Trust-based P2P overlay network for creating vehicle clusters. To simplify network management, a consistent hashing technique (chord) is employed, reducing complexity. Each Trust-based P2P group applies evidence theory and control flow to determine the optimal outcome for transmission to the global server. The proposed method also claims to achieve faster learning compared to convolutional federated learning.

3 Discussion

P2P Deep Learning really holds a great opportunity for Edge Computing. We have observed that being able to perform p2p deep learning in this field can provide us with various benefits. This approach enables IoT devices to collectively train models through peer-to-peer communication, leveraging their computational capabilities. In this way, edge devices will become smarter and the task of deep learning will be lifted from the central cloud alone. However, there are still various

difficulties in doing so. This is because although deep learning is an existing and well-researched field, performing it in edge computing environments, IoT devices or Fog layers has its own challenges.

Federated learning has been one of the issues we focused on in IoT edge computing. This method takes privacy and communication into account and does deep learning on IoT devices collaboratively without sharing raw data. In this way, data processing can be done in a decentralized way at the edge of the network. To fully utilize federated learning in IoT edge contexts, however, issues including diverse device capabilities, network connection, and data heterogeneity must be resolved. We mentioned the lack of standardization in IoT devices among the environmental problems.

Collaborative deep learning was another method we focused on in this regard. This strategy analyzes the trade-off between model accuracy and resource usage while concentrating on effective model aggregation. Of course, the fact that IoT devices are resource-constrained devices is one of the most important points that this method tries to pay attention to and overcome.

These deep learning methods can provide us with various advantages in the IoT environment. A Latif U. et al.(2020), study by Edge-Computing-Enabled Smart Cities shows us what deep learning can add[18]. By making decision-making processes smarter, it helps IoT devices in smart cities work more efficiently and well. Deep learning, for instance, can be used to forecast popularity and enable wise cache. Additionally, it can be applied to the efficient scheduling of mobility-aware caching and the creation of simple systems for commercial applications[18]. It can be used to optimize all areas in the city where IoT devices are used and integrated. We can pave the way for more efficient use of most areas, from car parks to the health sector, by deep learning in edge computing.

Besides these methods and benefits, how could we address the low-latency expectation and the resource-constrained edge computing area that makes it difficult to perform P2P deep learning on IoT devices? The coordination of edge and fog computing is one method for overcoming the resource limitations in edge computing. It is possible to increase computational benefits and improve resource usage by dividing computing and networking workloads among edge and fog resources[8]. However, we could not find any research that carried out deep learning with this orchestration. In our opinion, this fog edge can work together to benefit from increased computation power deep learning. Research is needed in this area.

Additionally, interesting findings have been obtained from research on latency-constrained deep neural network (DNN) architecture learning for edge systems. Deep learning can be carried out more quickly and with fewer resources by optimizing models for high accuracy while meeting low latency. The framework uses a machine learning-based latency predictor that allows for forecasts without the need for time-consuming device measurements[10].

Regarding data privacy and security in P2P deep learning, several methods were discussed in different contexts: federated learning, fog computing, and a private peer-to-peer network for distributed machine learning. Each method addresses the challenges of privacy preservation and introduces specific techniques to ensure the confidentiality of user data. While federated learning offers advantages in terms of privacy by keeping the training data decentralized, it is also vulnerable to adversarial attacks. This demonstrates a weakness in federated learning's security. Additionally, there is cause for concern that parameter adjustment could reveal information about a user's training data, compromising privacy. By employing MPC, the computation of model parameter averages can be done in a secure manner without revealing individual updates. However, this approach primarily focused on mobile devices and does not consider scenarios with limited communication resources.

In the context of private peer-to-peer networks for distributed machine learning, the authors claim that this method achieves faster learning compared to convolutional federated learning, but the specifics of privacy preservation within this architecture are not discussed in detail. A blockchain-based solution was proposed by Du et al. [20]. It discusses the challenges of transmitting private data about personal identity and financial account information to nearby servers for computation and the need for a computing resource trading and data-sharing framework or platform to motivate edge servers. While this proposed solution aids the privacy concerns across IoT devices, it requires a centralized training process. Alwarafy et al. [19] did a survey on security and privacy issues in edge computing assisting IoT devices. It highlights the risks associated with the massive growth of IoT devices and the corresponding huge data traffic generated at the edge of the network, which created additional burdens on the state-of-the-art centralized cloud computing paradigm due to bandwidth and resource scarcity. This literature survey reinstates the need of peer to peer deep learning for IoT devices.

4 Conclusion

This literature study describes how peer-to-peer deep learning can help enhance edge computing in IoT devices. It emphasizes the difficulties brought on by resource limitations and the demand for collaborative learning strategies to lessen these restrictions. In order to increase the functionality of IoT devices, this study investigates a number of techniques, including federated learning, peer-to-peer deep learning, and combined orchestration of edge and fog computing. The importance of edge computing in enabling intelligent processing at the network edge and the demand for standardization in IoT devices are both emphasized in the study above. With the help of this study, we contend that these constraints can be removed and resource efficiency increased by combining edge and fog computing. Moreover, several privacy methods were discussed that could help preserve users' privacy while training models using peer-to-peer deep learning. Overall, the work underscores the need for more research in this field and offers insightful information about how deep learning and edge computing are combined in IoT devices.

5 References

[1] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1655-1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[2] Šajina, Robert & Tankovic, Nikola & Ipsic, Ivo. (2022). Peer-to-peer deep learning with non-IID data. Expert Systems with Applications. 214. 119159. 10.1016/j.eswa.2022.119159.

[3] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li and H. Vincent Poor, "Federated Learning for Internet of Things: A Comprehensive Survey," in IEEE Communications Surveys & Tutorials, vol. 23, no. 3, pp. 1622-1658, thirdquarter 2021, doi: 10.1109/COMST.2021.3075439.

[4] P. Joshi, M. Hasanuzzaman, C. Thapa, H. Afli and T. Scully, "Enabling All In-Edge Deep Learning: A Literature Review," in IEEE Access, vol. 11, pp. 3431-3460, 2023, doi: 10.1109/ACCESS.2023.3234761.

[5] Zhang, Qingyang & Zhong, Hong & Shi, Weisong & Liu, Lu. (2021). A trusted and collaborative framework for deep learning in IoT. Computer Networks. 193. 108055. 10.1016/j.comnet.2021.108055.

[6] Li, He & Ota, Kaoru & Dong, Mianxiong. (2018). Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. IEEE Network. 32. 96-101. 10.1109/MNET.2018.1700202.

[7] Saurabh Bagchi, Muhammad-Bilal Siddiqui, Paul Wood, and Heng Zhang. 2019. Dependability in edge computing. Commun. ACM 63, 1 (January 2020), 58–66. https://doi.org/10.1145/3362068

[8] Cominardi, L., Abdullaziz, O.I., Antevski., K.,Chundrigar, S.B., Gdowski, R., Kuo, P.H., Mourad,A., Yen, L.H., Zabala, A. (2018). Opportunities and Challenges of Joint Edge and Fog Orchestration. In 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, Spain, 2018, pp. 344-349.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[10] Shuo Huai, Di Liu, Hao Kong, Weichen Liu, Ravi Subramaniam, Christian Makaya, Qian Lin, Latency-constrained DNN architecture learning for edge systems using zerorized batch normalization, Future Generation Computer Systems, Volume 142, 2023, Pages 314-327.

[11] Zheng, Z., Lyu, H., Zhang, Z., Niyato, D., & Kim, D. I. (2019). Towards collaborative deep learning in edge computing environments. IEEE Network, 33(5), 60-65.

[12] Briggs, C., Fan, Z., & Andras, P. (2020). A Review of Privacy-preserving Federated Learning for the Internet-of-Things. ArXiv. /abs/2004.11794

[13] E.Bagdasaryan, A.Veit, Y.Hua, D.Estrin, and V.Shmatikov, "How To Backdoor Federated Learning," arXiv.org, p. arXiv:1807.00459, Jul. 2018.

[14] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical Secure Aggregation for Privacy-Preserving Machine Learning. In: ACM SIGSAC conference on computer and communications security, pp 1175–1191

[15] Chen, M., Shlezinger, N., Poor, H. V., Eldar, Y. C., & Cui, S. (2021). Communication-efficient federated learning. Proceedings of the National Academy of Sciences, 118(17), e2024789118. https://doi.org/10.1073/pnas.2024789118

[16] 1.Gutiérrez, N., Rodríguez, E., Mus, S., Otero, B., Canal, R.: Privacy Preserving Deep Learning Framework in Fog Computing, https://link.springer.com/chapter/10.1007/978-3-030-64583-0_45. https://doi.org/10.1007/978-3-030-64583-0_45.

[17] J. M. Jeon and C. S. Hong, "Scalable Private P2P network for distributed and Hierarchical Machine Learning in VANETs," 2021 International Conference on Information Networking (ICOIN), Jeju Island, Korea (South), 2021, pp. 627-629, doi: 10.1109/ICOIN50884.2021.9333988.

[18]Khan, Latif U. , Yaqoob, Ibrar , Tran, Nguyen , Kazmi, S.M. , Nguyen Dang, Tri , Hong, Choong Seon. (2020). Edge-Computing-Enabled Smart Cities: A Comprehensive Survey. IEEE Internet of Things Journal. PP. 1-1. 10.1109/JIOT.2020.2987070.

[19] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider and M. Hamdi, "A Survey on Security and Privacy Issues in Edge-Computing-Assisted Internet of Things," in IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4004-4022, 15 March15, 2021, doi: 10.1109/JIOT.2020.3015432.

[20] Du, Yao & Wang, Zehua & Leung, Victor. (2021). Blockchain-Enabled Edge Intelligence for IoT: Background, Emerging Trends and Open Issues. Future Internet. 13. 48. 10.3390/fi13020048

Machine Learning and Deep Learning for the Internet of Things with Edge Computing in Smart City

Assignment 4a. Literature Study

Group 21

Zhuofan Mei
Universiteit van Amsterdam
zhuofan.mei@student.uva.nl
Pengju Ma
Universiteit van Amsterdam
pengju.ma@student.uva.nl
Lin Tian
Universiteit van Amsterdam
lin.tian@student.uva.nl

Abstract

In this paper, we discuss the machine learning and deep learning methods for the Internet of Things (IoT) with edge computing in smart city applications. Specifically, we analyze the implementation of these technologies in healthcare, traffic management, urban and rural planning and resource management in recent years, and compare the strengths and possible improvement perspectives of these studies. We also discuss future research directions and open issues in this emerging field.

1 Introduction

Machine learning and deep learning have revolutionized various fields, including the Internet of Things (IoT) and edge computing, contributing to the advancement of smart cities. In this era of rapid technological growth, cities are embracing intelligent systems to enhance efficiency, sustainability, and overall quality of life. The amalgamation of machine learning and deep learning techniques with IoT and edge computing offers unprecedented opportunities for developing innovative solutions to address the challenges faced by urban areas.

This article aims to explore the application of machine learning and deep learning methodologies in the context of smart cities, specifically focusing on the Internet of Things and edge computing. We will examine these technologies from various perspectives, delving into the theoretical foundations and their practical implementation in intelligent urban environments. By analyzing the methods used and their potential benefits, we can gain insights into how machine learning and deep learning contribute to the evolution of smart cities.

2 Search Query

```
(("machine learning" OR "ML") AND ("deep learning" OR "DL") AND ("internet
of things" OR "IoT") AND ("edge computing" OR "edge") AND ("smart city"
```

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

```
OR "healthcare" OR "environment" OR "urban planing" OR "resource")) AND (
LIMIT-TO ( SRCTYPE,"j" ) OR LIMIT-TO ( SRCTYPE,"p" ) ) AND ( LIMIT-TO (
SUBJAREA,COMP ) OR LIMIT-TO ( SUBJAREA,ENGI ) OR LIMIT-TO ( SUBJAREA,MATH
) ) AND ( LIMIT-TO ( PUBYEAR,2023) OR ( LIMIT-TO ( PUBYEAR,2022) OR
( LIMIT-TO ( PUBYEAR,2021) OR LIMIT-TO ( PUBYEAR,2020) OR LIMIT-TO (
PUBYEAR,2019) OR LIMIT-TO ( PUBYEAR,2018) OR LIMIT-TO ( PUBYEAR,2017) OR
LIMIT-TO ( PUBYEAR,2016) OR LIMIT-TO ( PUBYEAR,2015) ) AND ( LIMIT-TO (
LANGUAGE,English ) )
```

Here is some criteria when we search the relevant paper, the date of publishing is limited to 2015 to 2023.

3 Methodology

3.1 Machine Learning and Deep Learning

Machine learning is a field that addresses two fundamental questions: how to design computer systems that can improve automatically through experience, and what are the underlying statistical, computational, and information-theoretic principles governing learning systems[15]. The study of machine learning has both theoretical and practical significance, as it provides insights into fundamental scientific and engineering questions and has resulted in the development of practical software for various applications.

One of the most commonly used machine learning approaches is supervised learning, which involves training models using labelled data to make predictions. Examples of supervised learning systems include spam classifiers, face recognition systems, and medical diagnosis systems. In these systems, the training data consists of pairs of input-output (x, y) examples, and the goal is to produce accurate predictions (y*) in response to new input queries (x*).

On the other hand, unsupervised learning focuses on analyzing unlabeled data based on assumptions about the underlying structural properties. This can involve techniques such as algebraic, combinatorial, or probabilistic analysis of the data.

Deep learning (DL) has emerged as a powerful subset of machine learning that has revolutionized various fields, including computer vision, natural language processing, and speech recognition. DL has been actively utilized in many IoT applications in recent years[26]. DL has shown great potential in revolutionizing the Internet of Things (IoT) landscape by enabling intelligent and autonomous systems. DL techniques can be effectively applied to IoT devices, which generate massive amounts of data, to extract valuable insights, make real-time decisions, and enhance overall system performance. DL algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be deployed at the edge of the IoT network to process data locally, reducing latency and bandwidth requirements. This enables tasks such as real-time object detection, anomaly detection, predictive maintenance, and smart resource management. DL in IoT opens up opportunities for improved efficiency, reliability, and security, making it a promising technology for a wide range of applications, including smart cities, healthcare monitoring, environmental monitoring, and intelligent transportation systems.

3.2 Edge Computing

Edge computing is a paradigm change in which computer resources and data processing are moved closer to the data source, i.e., "the edge" of the network, rather of depending on a centralized cloud-based system. This architectural solution tackles numerous major issues that have arisen as a result of the increasing development of IoT devices, including latency, bandwidth utilization, and privacy [36].

Edge computing seeks to limit the quantity of data sent to the cloud for processing, analysis, and storage. This is accomplished by processing the data directly on the IoT devices or on local edge servers. Edge computing's concentrated data processing is ideal for applications that require real-time or near-real-time response, such as autonomous vehicles or remote health monitoring systems. The incorporation of machine learning and deep learning into edge computing systems improves the capabilities of IoT devices even further. For example, by placing ML/DL models on edge devices,

they can assess and act on data generated locally without requiring constant cloud access. By retaining critical data on-device, this integration can increase system responsiveness, conserve bandwidth, and improve privacy [42].

However, applying ML/DL at the edge presents a number of issues. Because of the limited processing capabilities on edge devices, running complicated and resource-intensive ML/DL models on these devices may be impractical. Furthermore, due to the decentralized nature of edge computing, effective solutions for distributed learning and model updates are required. Several works have proposed solutions to these challenges. For instance, Kang et al [16]. proposed Neurosurgeon, a system that partitions DL inference between the cloud and edge, minimizing latency and energy consumption. Similarly, Wang et al. introduced an adaptive learning system for edge computing, which leverages reinforcement learning to optimize model partitioning between the edge and the cloud [41].

3.3 Internet of Things

The Internet of Things (IoT) is a paradigm for connecting billions of smart devices that can communicate with each other with minimal human intervention. Among the many industries where the IoT has applications are smart cities, healthcare, and industry. However, there were some issues during the development of the IoT, such as limited resources and computing power and the corresponding security concerns [5]. Therefore, many IoT-related tasks had been introduced to machine learning and deep learning techniques, and the preliminary results are promising. For instance, to enhance the power supply of smart grids, Li et al. built an IoT-based deep learning system that automatically extracts features from captured data and eventually provides accurate estimates of future load values [23]. Deep learning would be crucial to the development of IoT services in the future due to its high efficiency in analyzing complex data with edge computing as mentioned in the earlier section, and Alsheikh et al. suggest a framework that blends deep learning methods with Apache Spark. The inference phase was carried out on a mobile device, while Apache Spark was set up on a cloud server to enable data training. It is possible to offload processing activities from the cloud using this two-tiered system, similar to edge computing [4].

Besides, due to its heterogeneous, dynamic, and distributed nature, the IoT also faces several security issues. As a result, the security risk of IoT systems is higher than that of other computing systems. Traditional security measures, such as encryption, authentication, and access control, are insufficient to protect IoT systems from a variety of threats and attacks. Additionally, IoT solutions produce a ton of useful data, serious privacy violations could happen if this data is not sent and evaluated in a secure manner [3]. In order to improve network performance and safeguard user privacy when uploading data, Li et al. introduced deep learning for IoT into the edge computing environment. They also proposed algorithms to maximize the number of tasks in the edge computing environment by taking into account the constrained service capacity of edge nodes [21]. Admittedly, smart cities are composed of and built by various IoT devices, and the deployment of technologies, deep learning and edge computing, can effectively help solve such problems and improve performance.

Application in Smart City

4 Healthcare and Well-being

Health and well-being are important aspects of smart city development, as smart cities aim to improve citizens' quality of life and health outcomes through the use of digital technology, data analytics and innovative solutions. Benefits include enhanced computational efficiency of medical sensors for better body monitoring, optimal allocation of transmitted data volumes and computational resources, and security of patient data. This section focuses on applications in this area.

At the initial stage, Chen et al. [12] and O'Shea et al. [29] emphasised the importance of machine learning and deep learning for disease prediction and accurate analysis of medical data for patient care and community services. Recognising that this can help in disease prediction even in the presence of incomplete data, they proposed latent factor models to fill the gaps left by the data, as well as CNN disease risk prediction models based on structured and unstructured real-world hospital data. Deep learning can characterise images to compensate for the lack of machine learning algorithms in this area and the occasional inaccuracy of mathematical models on real images. Also In terms of

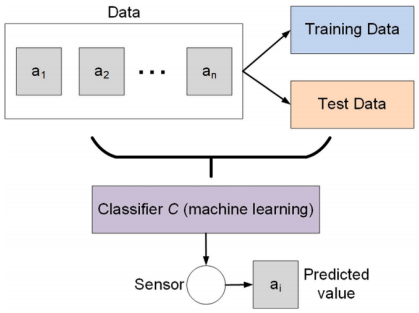


Figure 1: General application process

energy saving, Bassoli et al. [7] also proposed an intelligent plug-and-play system based on Wi-Fi connectivity that is suitable for body monitoring, which reduces the cost and difficulty of implementation by using an Internet connection based on a standard modem router and introduces a dedicated operating cycle that can save up to 91% of energy.

Regarding sensor-based applications. The general process of the machine learning algorithm implementation is shown in Figure 1, after the data is trained in the classifier, the trained classifier would assist the sensor in making appropriate judgments about the patient. Vippalapalli and Ananthula developed and implemented a prototype for a low-cost patient monitoring system based on a Body Sensor Network (BSN), which uses lightweight wearable sensors and sensor nodes to collect and analyse patient health data in real-time. These devices are designed to collect and share data, facilitating data storage and analysis while increasing productivity, meanwhile, they introduced a wearable device based on the Arduino Fio body sensor network platform for patient data collection, integrated with LabVIEW software to enable remote monitoring capabilities [40].

Nweke et al. [28] successfully analysed human physical activity and heart patterns using recursive neural algorithms, convolutional neural algorithm analysis, and data-intensive analysis, enabling sensitive body monitoring, but with problems related to high computational energy consumption and high error rates. To address this issue more effectively, condition-based monitoring has been implemented in multi-channel body monitoring systems [6]. This method would track a variety of body signals to detect abnormal activity in organs and other regions of the body at an early stage.

However, in some circumstances, network congestion between devices can lead to problems with the input and output of healthcare data and extra computation. Systems with IoT sensors provide healthcare data with much shorter response times and processing intervals to address network congestion and other issues due to potential interruptions or discontinuities in data transmission from remote physical monitoring systems [35]. Medical data privacy can also be compromised by interruptions in remote data processing intervals and potentially discontinuous data transmission. These led the researchers involved to introduce edge computing and in the early stages of applying this technology, Kumar et al. [19] suggested using an IoT protocol such as MQTT, deployed on medical endpoint IoT devices, to collect data from the cloud and perform offload operations (i.e. there is an MQTT client on the IoT device and an MQTT server on the edge device that can request other network services from the cloud) while considering multiple IoT edge servers, their interactions, and endpoint devices, which would be helpful to address the network latency caused by cloud computing.

With the development of methodology, Aazam et al. [1] provided various forms of intelligent and opportunistic healthcare by leveraging machine learning-based approaches to edge computing, applying K-nearest neighbour, simple Bayesian and support vector classification algorithms to real-world data tracking in healthcare and security scenarios, analysing indoor and live scenes, and implementing machine learning-based task offloading for edge computing, as a result, this technique improved efficiency and reduces the required computational resources. Besides, Manogaran et al. [25] designed and implemented wearable smart log patches with IoT sensors and deployed them in a multi-channel body monitoring system. The system uses multimedia technology, edge computing, agile learning, IoT sensors and real-time body data analysis. It also combines these technologies with Bayesian networks to track users' physical activity in real-time and process wearable IoT data. Tests show that edge computing with EC Bayesian Neural Networks (EC-BNN) can infer and iden-

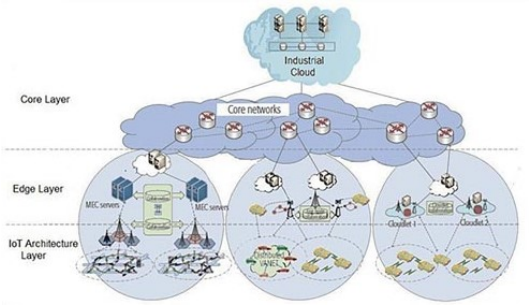


Figure 2: *(Source: J. Li et al., "A Secured Framework for SDN-Based Edge Computing in IoT-Enabled Healthcare System," in IEEE Access, vol. 8, pp. 135479-135490, 2020, doi: 10.1109/ACCESS.2020.3011503.)*

tify collected human body data and classify health and physical activity data in a highly predictive manner. Therefore, it could be applied to real medical analysis scenarios, with promising potential for improvement. These applications were also very helpful during the COVID-19 period, Lydia et al. [20] implemented a federated deep learning-based FDL-COVID detection model running on an IoT edge computing platform. The SqueezeNet model is used to create DL models from patient data captured by IoT devices, and servers use encrypted variables obtained from IoT devices to create the final global cloud model. And, Kong et al. [17] proposed an edge computing-based mask (ECMask) identification framework to support public health prevention and ensured the real-time functionality of low-power camera devices.

Rajavel et al. [31] proposed a cloud-based intelligent medical object tracking and behaviour recognition system (COTBIS) with a framework that combines moving object combination using improved background subtraction and deep CNN algorithms, abnormal fall activity monitoring detection and classification using hierarchical voting, and edge computing capabilities at the gateway level. This minimized network traffic and response times while increasing the accuracy of fall behaviour prediction. In terms of security, Li et al. [22] proposed a software-defined network (SDN) security framework for edge computing in healthcare systems. Patient data would be sent to the edge server for storage, processing and analysis after IoT devices are authenticated by it using a lightweight authentication scheme. SDN-based edge computing can then perform better load balancing, network optimisation and efficient resource utilisation through optimised network configuration. The framework of SDN-based Edge Computing is shown in Figure 2.

5 Intelligent Transportation System

5.1 Route Planning Algorithms

Route planning is important in the context of intelligent transportation systems. The demand for effective, affordable, and ecological transportation options is greater than ever as urban populations rise and automobile use rises. Choosing the most effective route for a trip is a process known as route optimization, and it is without a doubt the hottest topic in modern intelligent transportation. Along with convenience, it has an impact on a number of social factors, such as lowering fuel usage, lowering carbon emissions, and enhancing overall transportation effectiveness. Delivery firms can guarantee timely and economical deliveries with the aid of smart route optimization, emergency services can reach their locations more quickly, and commuters may cut down on travel time and enhance their travel experience.

In terms of algorithms, we can use this model to think about urban traffic network: the overall urban traffic network consists of countless nodes and edges, where intersections are equivalent to nodes and the roads connected between intersections form edges. Traditional graph algorithms such as Dijkstra's algorithm can calculate the shortest path based on the weight of each edge [34], however, the real-world traffic network is not static, such as there are special circumstances such as road construction and destruction, traffic control, etc. In addition, there are other factors such as user preferences, for example, some users tend to spend more time to go farther to bypass the weak

safety control in the city, and there are other factors such as traffic costs, etc. The application of traditional algorithms in multi-factor dynamic traffic networks is extremely limited.

Traffic forecasting has frequently employed supervised learning. For example, regression models have been used to forecast journey times, and classification models can be used to foretell whether a route would be congested or free-flowing. Pfeiffer, Mark, et al. [30] proposed supervised learning approach uses CNN to analyze and process the laser information collected by sensor devices, and uses the classical A* algorithm as the training marker information. Nevertheless, despite Supervised Learning's success in these fields, it still faces substantial obstacles in its application to dynamic route optimization. For example, the approach mentioned above relies strongly on the tagging algorithm. In essence, Supervised Learning mainly relies on fixed labels and past data to train the model. This is a drawback in dynamic traffic settings when conditions are ever-changing and past patterns might not be a reliable indicator of present ones. Furthermore, Supervised Learning is unable to draw lessons from novel circumstances that did not exist in the training data.

This is where Reinforcement Learning (RL) comes in. RL is a subset of Machine Learning that enables learning from the results of decisions rather than relying exclusively on past data [37]. Route planning is a standard MDP(markov decision processes) problem [39] in the context of reinforcement learning scenarios, using value iteration methods [38] to derive mapping information between all location states and specific action decisions. In this way, the next movement path can be quickly determined regardless of the current location in the traffic network. An RL agent can continually interact with the dynamic traffic environment, make decisions, and learn from the rewards or punishments it receives in the context of route optimization. The use of RL for route optimization offers a possible countermeasure to the drawbacks of supervised learning. RL can adapt to new circumstances that weren't present in the training data, unlike supervised learning. This is particularly useful in situations with dynamic traffic, when unforeseen occurrences like accidents, construction zones, or abrupt changes in the weather are possible. Although RL offers these important benefits, it is not without difficulties. In order to learn properly, RL often necessitates several interactions with the environment, which can be time-consuming and computationally costly.

5.2 System Architecture

Intelligent Transportation Systems (ITS)'s (ML) and Deep Learning (DL) models must be deployed and used in a way that is compatible with the underlying software architecture for them to be effective. Frameworks for data processing and storage that are reliable and scalable are required due to the complexity and volume of data's growth. Distributed computing platforms like Hadoop's Hadoop Distributed File System (HDFS) and Apache Spark stand out as crucial technologies in this setting.

Massive volumes of data collected from numerous sources, including sensors, GPS systems, traffic cameras, and linked vehicles, are stored using HDFS, which is essential to ITS. Due to HDFS's distributed architecture, it can store and retrieve huge data sets over numerous nodes while yet maintaining data availability and fault tolerance. This is especially important in ITS, where data loss can result in serious operational inefficiencies [9]. Apache Spark, on the other hand, provides an efficient distributed processing solution for data analysis and ML/DL model training on large-scale data stored in HDFS. In the context of ITS, Spark can help to speed up activities like traffic pattern analysis, congestion prediction, and route optimization by allowing for fast, distributed processing of big datasets and efficient training and deployment of ML/DL models. Spark's MLlib, for example, may be used to train a deep learning model on a huge corpus of traffic data stored in HDFS, allowing the system to forecast future traffic conditions and optimize traffic flow [43]. Furthermore, for real-time traffic management, these tools can be integrated with real-time data processing frameworks such as Apache Kafka and Apache Flink. Kafka can receive real-time data from various traffic sensors and devices, which Flink can then use for real-time analytics and ML/DL model inference. For example, on Flink, a machine learning model can be implemented to analyze real-time data from Kafka, detecting possible issues or congestions and delivering rapid input to the traffic management system [18].

To summarize, the utilization of distributed storage and processing technologies such as HDFS and Spark, as well as real-time data processing systems like as Kafka and Flink, enables ITS to manage massive volumes of data and deploy ML/DL models for both historical data analysis and real-time

decision-making. This combination of big data technology and machine learning/deep learning models provides the door for more intelligent, efficient, and responsive transportation systems.

6 Urban Planning and Resource Optimization

6.1 Land Use Planning

Land use planning is an important urban planning activity that aims to rationalize different uses of land for sustainable urban development and optimal spatial utilization. Through land use planning, urban planners and policy makers can determine the best use of land, including residential, commercial, industrial, agricultural, and nature conservation areas, to meet people's needs for housing, employment, transportation, and greenery.

Land use planning takes into account several factors, including population growth trends, economic development needs, environmental protection, and social equity. By analyzing and evaluating the city's existing land resources, land use, citizens' needs and future development trends, appropriate planning strategies and goals are formulated to guide the development and use of land.

Advanced technologies such as machine learning and deep learning are increasingly being applied in the land use planning process. These technologies are able to process and analyze large amounts of geographic data, remotely sensed imagery, and spatial information to provide more accurate land use classification and change simulation predictions. Machine learning algorithms such as support vector machines and random forests are able to classify and predict based on known land use samples, helping planners understand the characteristics and trends of different land types. In contrast, deep learning methods such as convolutional neural networks are able to learn and extract complex features of land use from large-scale data, providing more comprehensive information for planning decisions.

Support Vector Machine (SVM) is a machine learning algorithm widely used for land use classification. It is designed to handle high-dimensional data and nonlinear relationships to classify different classes of land use by constructing a separating hyperplane in the feature space.

The basic idea of SVM is to find an optimal hyperplane that correctly separates the different classes of land use samples. The selection of the hyperplane is based on the attributes of the training samples and their distribution in the feature space. Specifically, SVM determines the optimal hyperplane by maximizing the spacing between the data points and the decision boundary, which is called the maximum spacing method[10].

In conclusion, the support vector machine is a powerful machine learning algorithm that plays an important role in land use planning. It can effectively handle high-dimensional data and nonlinear relationships, and improve the accuracy of classification by controlling the overfitting problem. By applying SVM, land use identification, classification and prediction can be achieved, providing an effective tool and method for land use planning.

The performance of the Random Forest (RF) algorithm was evaluated in a study on land use classification in the province of Granada, Southern Spain, using multitemporal Landsat-5 thematic mapper data (Rodriguez-Galiano et al,2012) . The study compared the performance of RF with conventional classification trees (CT) and concluded that RF provided more significant differentiation of land cover categories.

Another study adopted a hybrid approach for land use classification, combining the object-oriented method with deep Convolutional Neural Network (CNN) known as COCNN. The object-oriented method was utilized to construct a multiscale sample set, providing high precision training data. Remote sensing images covering an area around Fuxian Lake were used to classify ten land use types.

In the modified approach, convolution kernels, which directly extract low-level features from the original image, were considered the most sensitive element of the CNN. The results of the hybrid approach were compared with those obtained from a standalone CNN, and it was observed that the modified approach yielded improved results. The paper concluded that the choice of kernel size played a crucial role in the outcome, and a kernel size of 3x3 was used for the object-oriented training sets[10].

Deep learning methods, such as Convolutional Neural Network (CNN) and Artificial Neural Network (ANN), also have important applications in land use planning. Deep learning methods perform feature learning and representation learning through a multilayer neural network structure, and are capable of handling large-scale geographic and image data. For example, in land use classification tasks, CNNs can extract spatial and texture features from satellite images to achieve accurate classification of different land use types.

Water resource management is a notoriously difficult problem, mainly because of the uncertainty regarding inflows to the system, thereby making inevitable the probabilistic approach, which accordingly sets a framework of policy decisions related to the acceptable risk of failure[33].

6.2 Water Resources Management

Machine learning can be used for prediction and optimization of water resources systems. By analyzing and learning from historical data, machine learning models can predict trends and probability distributions of future hydrologic variables (e.g., rainfall, water levels, runoff, etc.). This provides important information for water resources planning and decision making. In addition, machine learning can help determine the best water resource allocation and utilization options through optimization algorithms to improve water resource utilization efficiency and system performance[14].

Plus, these techniques can be applied to decision support systems in water resources management. By building sophisticated machine learning models and algorithms, water resources systems can be simulated and optimized to support decision makers in making informed decisions. These decisions may involve reservoir scheduling, water allocation, irrigation management, etc. Machine learning models can consider multiple factors and constraints to provide feasible solutions and decision recommendations.

In addition, for anomaly detection, and troubleshooting in water resources management. By monitoring the operational data of water resources systems, machine learning models can identify abnormal events and faults and provide real-time alerts and recommendations. This helps to take timely action to respond to problems and ensure the stable operation of water resources systems.

However, there are some challenges because of the edge computing combining with the machine learning and deep learning.

1.Data quality and availability: there are often problems with missing, incomplete or inaccurate data. This can affect the accuracy and reliability of machine learning and deep learning algorithms. Ensuring the quality and availability of data is therefore an important challenge.

2.Data privacy and security: Smart cities involve a large amount of data collection and sharing. These data often contain personal privacy and sensitive information. Therefore, ensuring data privacy and security when processing such data using machine learning and deep learning algorithms is an important challenge. Appropriate data protection and security measures must be taken to prevent data leakage and misuse.

3.Algorithmic and computational resource limitations: Therefore, computational resource limitations need to be addressed when applying these algorithms in smart cities. It may be necessary to optimize the algorithms, use distributed computing, or select appropriate hardware to overcome these limitations.

4.Interpretability and transparency: Machine learning and deep learning algorithms are often presented in a black box format, making it difficult to explain their decision processes. Transparency and interpretability are crucial so that decision makers can understand the basis of the algorithms' decisions. Therefore, improving the interpretability of machine learning and deep learning algorithms is a challenge[11].

7 Discussion

Edge computing allows data to be processed closer to its source, reducing latency and bandwidth consumption, while leveraging machine learning and deep learning to solve as many challenges as possible in Smart City applications. Specifically, the heterogeneity of edge devices, data security and privacy, the scalability and reliability of edge networks, and the complexity and efficiency of

machine learning and deep learning models are some of the challenges and limitations that need to be further overcome [8], and while these issues have been addressed upfront to the extent possible, there is still room for improvement, some efforts made are shown below.

- Develop distributed, adaptive machine learning and deep learning algorithms that can operate on a variety of heterogeneous edge devices with limited hardware and software. Luo et al. [24] proposed the ProbComp-LPAC algorithm to implement probabilistic compression and adaptive compression of layer parameters to reduce communication costs and increase the training efficiency of distributed deep learning. The algorithm uses probabilistic equations to select gradients. It uses different compression rates in different layers of the deep neural network, resulting in state-of-the-art performance compared to other approaches in the same time frame.

- Implement security and privacy mechanisms to protect against unauthorised access and manipulation of data and models at the edge. The Multi-Access Edge Computing (MEC) paradigm was previously introduced by the European Telecommunications Standards Institute to enable efficient and fast data processing in mobile networks. Ranaweera [32] examined the security and privacy vulnerabilities of MEC systems and the threats they pose. They also identified and investigated the threat vectors, proposed potential security solutions and focused on the privacy concerns of MEC systems.

- Improve resource allocation, fault tolerance, load balancing and network slicing capabilities of edge networks to improve their scalability and reliability. And use methods such as model compression, pruning, quantization and refinement to optimize the complexity and effectiveness of machine learning and deep learning models. There are already some applications using federated learning, which improves the security of IoT applications at the edge, but how to deal with the communication load it imposes is an issue that needs to be further explored in the future [2].

Also, we found there were some research and work based on fog computing [27]. Fog computing works by inserting a processing layer between the edge device and the cloud, filtering out unnecessary data and sending only relevant data to the cloud for further processing or storage. It can offer advantages over edge computing in terms of scalability, security and reliability, and can handle large amounts of data from multiple sources, providing backup and recovery options in the event of network failure [13]. However, edge computing offers lower latency and lower network bandwidth consumption by bringing computing closer to the data source. Overall, the decision between fog and edge computing is influenced by a number of variables, including the depth of the deep learning model, the volume of data to be processed, the computing capabilities of the device, and the required response time.

8 Conclusions

We delved into the nuanced applications of machine learning and deep learning within the ambit of smart cities. We particularly emphasized their interplay with the Internet of Things (IoT) and edge computing. This investigation allowed us to appreciate how these advanced technologies drive the smart city revolution, enabling us to address pressing urban challenges with data-driven insights and automated solutions.

Machine learning and deep learning, we discovered, serve as potent solutions to numerous urban dilemmas, including traffic congestion, energy efficiency, and waste management. When coupled with IoT and edge computing, they form a formidable alliance that powers innovative solutions, elevating urban efficiency, sustainability, and the overall quality of life. We further expanded on real-world implementations of these technologies, providing tangible examples such as health care and well-being, intelligent transportation systems and efficient resource management.

Our literature review highlights the critical significance of machine learning and deep learning in shaping smart city architecture. These aren't just tools; they're the engines that are transforming cities into technologically advanced, data-driven, and responsive ecosystems. As we continue to harness and enhance these technologies, the future of smart cities holds limitless possibilities. Anticipating these future advancements excites us and motivates us to continue exploring, trying, and developing.

References

[1] Mohammad Aazam, Sherali Zeadally, and Eduardo Feo Flushing. Task offloading in edge computing for machine learning-based smart healthcare. *Computer Networks*, 191:108019, 2021.

[2] Haftay Gebreslasie Abreha, Mohammad Hayajneh, and Mohamed Adel Serhani. Federated learning in edge computing: a systematic survey. *Sensors*, 22(2):450, 2022.

[3] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Khalid Al-Ali, Xiaojiang Du, Ihsan Ali, and Mohsen Guizani. A survey of machine and deep learning methods for internet of things (iot) security. *IEEE Communications Surveys & Tutorials*, 22(3):1646–1685, 2020.

[4] Mohammad Abu Alsheikh, Dusit Niyato, Shaowei Lin, Hwee-Pink Tan, and Zhu Han. Mobile big data analytics using deep learning and apache spark. *IEEE network*, 30(3):22–29, 2016.

[5] Hamidreza Arasteh, Vahid Hosseinnezhad, Vincenzo Loia, Aurelio Tommasetti, Orlando Troisi, Miadreza Shafie-khah, and Pierluigi Siano. Iot-based smart cities: A survey. In *2016 IEEE 16th international conference on environment and electrical engineering (EEEIC)*, pages 1–6. IEEE, 2016.

[6] Yuequan Bao, Zhiyi Tang, Hui Li, and Yufeng Zhang. Computer vision and deep learning–based data anomaly detection method for structural health monitoring. *Structural Health Monitoring*, 18(2):401–421, 2019.

[7] Marco Bassoli, Valentina Bianchi, and Ilaria De Munari. A plug and play iot wi-fi smart home system for human monitoring. *Electronics*, 7(9):200, 2018.

[8] Pierfrancesco Bellini, Paolo Nesi, and Gianni Pantaleo. Iot-enabled smart cities: A review of concepts, frameworks and key technologies. *Applied Sciences*, 12(3):1607, 2022.

[9] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11(2007):21, 2007.

[10] Vineet Chaturvedi and Walter T de Vries. Machine learning algorithms for urban land use planning: A review. *Urban Science*, 5(3):68, 2021.

[11] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[12] Min Chen, Yixue Hao, Kai Hwang, Lu Wang, and Lin Wang. Disease prediction by machine learning over big data from healthcare communities. *Ieee Access*, 5:8869–8879, 2017.

[13] Koustabh Dolui and Soumya Kanti Datta. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2017.

[14] Jeffery S Horsburgh, David G Tarboton, David R Maidment, and Ilya Zaslavsky. A relational model for environmental and water resources data. *Water Resources Research*, 44(5), 2008.

[15] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[16] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[17] Xiangjie Kong, Kailai Wang, Shupeng Wang, Xiaojie Wang, Xin Jiang, Yi Guo, Guojiang Shen, Xin Chen, and Qichao Ni. Real-time mask identification for covid-19: An edge-computing-based deep learning framework. *IEEE Internet of Things Journal*, 8(21):15929–15938, 2021.

[18] Jay Kreps, Neha Narkhede, Jun Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, pages 1–7. Athens, Greece, 2011.

[19] S Mohan Kumar and Darpan Majumder. Healthcare solution based on machine learning applications in iot and edge computing. *International Journal of Pure and Applied Mathematics*, 119(16):1473–1484, 2018.

[20] E Laxmi Lydia, CSS Anupama, A Beno, Mohamed Elhoseny, Mohammad Dahman Alshehri, and Mahmoud M Selim. Cognitive computing-based covid-19 detection on internet of things-enabled edge computing environment. *Soft Computing*, pages 1–12, 2021.

[21] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network*, 32(1):96–101, 2018.

[22] Junxia Li, Jinjin Cai, Fazlullah Khan, Ateeq Ur Rehman, Venki Balasubramaniam, Jiangfeng Sun, and P. Venu. A secured framework for sdn-based edge computing in iot-enabled healthcare system. *IEEE Access*, 8:135479–135490, 2020.

[23] Liangzhi Li, Kaoru Ota, and Mianxiong Dong. When weather matters: Iot-based electrical load forecasting for smart grid. *IEEE Communications Magazine*, 55(10):46–51, 2017.

[24] Peng Luo, F. Richard Yu, Jianyong Chen, Jianqiang Li, and Victor C. M. Leung. A novel adaptive gradient compression scheme: Reducing the communication overhead for distributed deep learning in the internet of things. *IEEE Internet of Things Journal*, 8(14):11476–11486, 2021.

[25] Gunasekaran Manogaran, P Mohamed Shakeel, Hassan Fouad, Yunyoung Nam, S Baskar, Naveen Chilamkurti, and Revathi Sundarasekar. Wearable iot smart-log patch: An edge computing-based bayesian deep learning network system for multi access physical monitoring system. *Sensors*, 19(13):3030, 2019.

[26] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018.

[27] Ammar Awad Mutlag, Mohd Khanapi Abd Ghani, Net al Arunkumar, Mazin Abed Mohammed, and Othman Mohd. Enabling technologies for fog computing in healthcare iot systems. *Future Generation Computer Systems*, 90:62–78, 2019.

[28] Henry Friday Nweke, Ying Wah Teh, Ghulam Mujtaba, and Mohammed Ali Al-Garadi. Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions. *Information Fusion*, 46:147–170, 2019.

[29] Timothy Oshea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[30] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017.

[31] Rajkumar Rajavel, Sathish Kumar Ravichandran, Karthikeyan Harimoorthy, Partheeban Nagappan, and Kanagachidambaresan Ramasubramanian Gobichettipalayam. Iot-based smart healthcare video surveillance system using edge computing. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13, 2022.

[32] Pasika Ranaweera, Anca Delia Jurcut, and Madhusanka Liyanage. Survey on multi-access edge computing security and privacy. *IEEE Communications Surveys Tutorials*, 23(2):1078–1124, 2021.

[33] Evangelos Rozos. Machine learning, urban water resources management and operating policy. *Resources*, 8(4):173, 2019.

[34] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM J. Exp. Algorithmics*, 5:12es, dec 2001.

[35] P Mohamed Shakeel and Gunasekaran Manogaran. Prostate cancer classification from prostate biomedical data using ant rough set algorithm with radial trained extreme learning neural network. *Health and Technology*, 10:157–165, 2020.

[36] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[37] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[38] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

[39] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. *Reinforcement learning: State-of-the-art*, pages 3–42, 2012.

[40] Vikas Vippalapalli and Snigdha Ananthula. Internet of things (iot) based smart health care system. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pages 1229–1233, 2016.

[41] Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, and Wenbo Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *Ieee Access*, 5:6757–6779, 2017.

[42] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *Wireless Algorithms, Systems, and Applications: 10th International Conference, WASA 2015, Qufu, China, August 10-12, 2015, Proceedings 10*, pages 685–695. Springer, 2015.

[43] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

What are the emerging technologies being adopted in cloud computing to address the privacy and data protection requirements for the GDPR-specific challenges?

Group 21

Abstract

This literature research explores the challenges of GDPR compliance in cloud computing and investigates technologies and techniques to address them. It focuses on data retention, data portability, and visibility of metadata and data minimization. This research discusses approaches such as data classification, encryption, interoperability standards, and data anonymization. It highlights the importance of GDPR compliance for cloud service providers and offers insights into existing and emerging solutions. Overall, it provides valuable guidance for organizations aiming to protect personal data in the cloud while adhering to GDPR requirements.

1 Introduction

In recent decades, cloud services are getting popular deployment in various business sectors which also led to data leaks and privacy issues. The previous research of Arcserve [3] stated that there are seven infamous cloud security breaches. To name a few, 530 million Facebook users' personal data were breached around August 2019, in November of 2019 1.1 billion user data from Alibaba were illegally scraped, and in 2021 LinkedIn also fell victim to the data breach. As the cases show, this issue is not only bounded to small firms but also to large tech companies.

GDPR compliance data protection in the cloud is an extensive research field with various protection layers to be considered, Takabi et al. [20]. As a result, it is difficult to derive a single solution that fulfills all the requirements of the GDPR. In a previous study, regarding the GDPR impact on cloud computing, conducted by Tolsma [21], nine main GDPR-specific challenges were proposed. The first challenge is implementing retention effectively in the cloud which comprises identifying and managing multi-jurisdictional retention requirements, securing backup, and managing data deletion. Second, data portability for the controller denotes the technical ability to ensure the data subject rights. Third, the visibility and minimization of metadata illustrate control on intended use/access and level of protection on the metadata. The remaining six challenges are, namely, response to the breach, strategy to process personal data outside of the European Economic Area, risk management of the cloud provider, monitoring architecture of the cloud provider's system to be alerted of changes and updates of the system, assessment of IT security requirements, and control over the data ownership.

In this paper review, we will concentrate on how existing or emerging technologies and data protection techniques can adhere to the three GDPR-specific challenges: implementing retention effectively in the cloud, data portability for the controller, and visibility regarding metadata and data minimization. Section 2 will be dedicated to the inspection of prior research that introduced possible techniques that deal with the proposed three challenges and the weaknesses of the proposed solution. We will explore solutions to the challenges starting from retention implementation in the cloud followed by data portability, and visibility and minimization of metadata. In Section 3, a general conclusion of our review will be provided with future research directions in terms of GDPR-specific data protection in the cloud.

2 Three GDPR-specific challenges

In this section, we will delve into how existing data protection technologies can solve or comply with the three GDPR-specific challenges in cloud computing: implementing retention effectively in the cloud, data portability for the controller, and visibility regarding metadata and Data Minimization. Furthermore, we will highlight the strengths and weaknesses of these underlying protection solutions.

2.1 Implementing retention effectively in the cloud

Under the GDPR, personal data should not be stored for longer than necessary for the intended purpose, stated by Politou et al. [16]. This requires implementing retention periods and effectively deleting data when those retention periods expire, according to Rhahla et al. [18]. However, managing retention and the deletion of data becomes challenging due to data being stored across multiple locations and jurisdictions by cloud service providers. Additionally, deleting data completely involves considering backups, which makes the process even more complex. Therefore, it is crucial to have a clear understanding of how cloud service providers secure backups and manage data retention. To ensure compliance with GDPR privacy and data protection requirements, emerging technologies and practices in cloud computing are being adopted to address the challenge of effective data retention in the cloud. These approaches aim to manage retention periods, facilitate data deletion, and handle multi-jurisdictional retention requirements.

Approaches for retention challenges

Data Classification and Tagging: Cloud service providers use data classification and tagging to organize and label data based on its sensitivity, purpose, and retention requirements which helps organizations manage and track data in the cloud as stated by, Politou et al. [15]. Abiteboul and Stoyanovich [1], argue that by categorizing data and assigning specific tags (tagging), organizations can improve their management regarding retaining and deleting data, ensuring compliance with the regulations of GDPR.

Wang and Shah [23], shows that a specific technology in the field of classification and data tagging for data retention is the use of metadata-based tagging systems. These systems use metadata, such as data attributes, labels, or keywords, to categorize and tag data based on its sensitivity, purpose, and retention requirements. According to Politou et al. [15], by implementing metadata-driven tagging mechanisms, organizations can manage and track data, and therefore be able to ensure compliance with GDPR's privacy and data protection requirements.

Encryption and Tokenization: Cloud providers are using encryption and tokenization techniques to improve data security and privacy. These technologies can help protect data while being stored and transferred, ensuring that even if data is retained for longer periods, it remains secure and unreadable by unauthorized parties, as stated by Rhahla et al. [18]

An emerging technology in the field of encryption and tokenization is Differential Privacy. Differential Privacy is a technique that adds random or noisy data to datasets to protect the privacy of individuals, according to Holzel [9]. It helps ensure that personal information stays private while still allowing for insights to be drawn from the combined data. Because of the randomness, it becomes more difficult to identify specific individuals within the dataset, thus ensuring their privacy, as shown by Holzel [9]. Differential Privacy is gaining attention and being explored as a promising approach to address privacy concerns and comply with GDPR's privacy and data protection requirements.

Data Lifecycle Management: Cloud platforms include Data Lifecycle Management (DLM) features for better data control. This includes automated data expiration, options for long-term storage, and policy-driven data deletion. These capabilities help organizations in complying with retention periods and effectively manage data, even across multiple jurisdictions, as proposed by Rahul and Banyal [17].

DLM plays a necessary role in the compliance of GDPR requirements regarding data retention. According to Rahul and Banyal [17] it enables organizations to set up and enforce retention periods for personal data, ensuring that data is not stored longer than necessary. By implementing DLM features, organizations can manage data throughout its lifecycle, which includes correct archival and deletion when retention periods expire. This helps organizations meet GDPR's data retention obligations, protect individuals' privacy rights, and demonstrate compliance with regulatory requirements.

Geographically Distributed Data Storage: Cloud service providers (CSP) are offering options for geographically distributed data storage. According to Zhou et al. [27], CSPs allow organizations to choose data centers located in specific regions where their data will be stored. This ensures that data is kept within the desired jurisdiction and complies with local data protection laws. Based on Vogt et al. [22], cloud providers have multiple data centers within a region, known as availability zones. Organizations can select specific availability zones to store their data and ensure services and data remain accessible even if one zone experiences issues. It improves reliability and provides compliance with multi-jurisdictional retention requirements. As mentioned in Abualkishik et al. [2], cloud providers offer replication services that automatically duplicate data across different regions. This helps organizations meet compliance requirements by ensuring data is stored in multiple locations and facilitating disaster recovery. Cloud providers offer data sovereignty options that allow organizations to store data exclusively within a specific country or region. This helps ensure compliance with local data protection and privacy laws, as mentioned in Kushwaha et al. [11].

Transparent Data Deletion Processes: Based on Abiteboul and Stoyanovich [1], cloud providers are creating clear and open procedures for deleting data, giving organizations and individuals a better understanding of how data is removed and how backups are managed. This involves establishing guidelines for data deletion, keeping track of actions taken in this process through audit trails, and offering tools or interfaces that allow users to request data deletion.

To ensure accountability, transparency, and the ability to track and verify data processing. ledgers, distributed ledger technology, can be used and refers to a transparent and unchangeable record that documents all data transactions and activities related to data processing, as mentioned in Bonatti et al. [5] and Kuperberg [10]. As stated by Bonatti et al. [5], it serves as a broad log of who shared data, with whom, and for what purpose. The ledger captures important information such as data collection, storage, processing, transmission, and any other relevant operations. It enables individuals, to have insights into how their data is being handled and allows companies to be and show their compliance with GDPR regulations, according to Kuperberg [10].

2.2 Data portability for the controller

Several cutting-edge techniques and technologies have been used to fulfill the General Data Protection Regulation's (GDPR) requirement for data portability in the context of cloud computing. With these developments, personal data can be seamlessly transferred between different cloud service providers or controllers while maintaining privacy and data protection. Here are a few crucial strategies being used:

Interoperability Standards: Data portability is made easier by the creation and application of interoperability standards, which make it possible for data to be seamlessly transferred between various cloud platforms. In order to extract and transport personal data in a consistent manner, controllers might use these standards, which define common data formats, protocols, and interfaces. For example XML and JSON formats. XML (Extensible Markup Language) and JSON (JavaScript Object Notation) are widely used formats for structured data. They provide a common syntax and structure that can be understood by different systems and programming languages. XML has been a longstanding standard, while JSON has gained popularity due to its simplicity and compatibility with JavaScript-based applications according to De Hert et al. [7]. A weakness of this technique is that not all systems or platforms may fully adhere to the same interoperability standards. So in essence this method can only work perfectly if all data controllers are on the same page when it comes to interoperability standards. A slight variation in the interpretation of standards could hinder seamless integration and data exchange between different systems.

Application Programming Interfaces(APIs): More and more cloud service providers are providing APIs that let controllers access and retrieve personal data from their platforms using programming. These APIs offer a safe and consistent mechanism to retrieve data, enabling controllers to quickly

satisfy data portability needs according to Petcu [14]. An example of such a technology is the RESTful API. RESTful APIs provide a standardized and lightweight approach to building APIs that can be easily consumed by different systems and programming languages. They typically use common HTTP methods (such as GET, POST, PUT, and DELETE) for data retrieval, creation, updating, and deletion. A flaw of this approach is that APIs may change over time, introducing compatibility issues or requiring periodic modifications. Developers need to stay up-to-date with API changes and plan for potential disruptions during upgrades or API deprecations. Developers need to also be aware of unauthorized access, injection attacks, or exposure of sensitive data that can occur if API security measures are not implemented effectively.

Containerization and Virtualization: Applications and their dependencies can be encapsulated using virtualization techniques and containerization technologies like Docker and Kubernetes, making them portable across various cloud environments by Pahl [13]. These technologies enable controllers to bundle their applications and related data, facilitating data portability and facilitating easier migration between cloud providers. A downside of this technology is the performance overhead that can occur from the additional layers of abstraction that containerization and virtualization introduce. Running applications directly on bare-metal infrastructure can be more efficient.

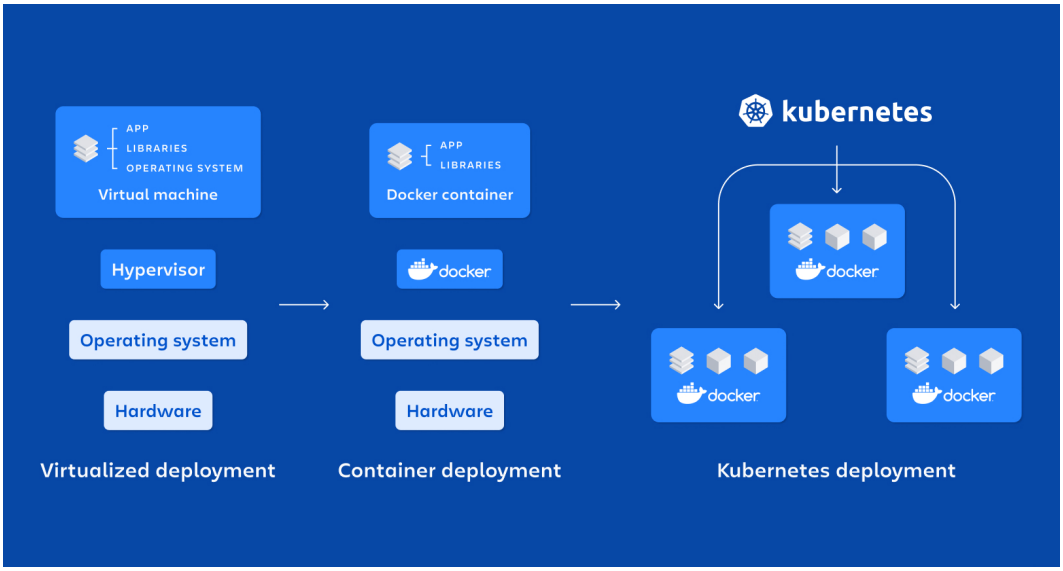


Figure 1: Containerization using Docker and Kubernetes.

Data Minimization and Anonymization: Cloud service providers are implementing techniques like data minimization and anonymization in order to reduce the hazards connected with data portability. Data minimization entails storing or processing less personal data and decreasing the amount of data that must be transferred. Anonymization techniques reduce privacy risks throughout the portability process by converting personal data into a format that prevents individual identification.

Secure Data Transfer Mechanisms: Strong authentication systems, reliable encryption techniques, and secure data transfer channels are essential elements for assuring the secure transfer of personal data during the portability process. multiple encryption techniques and secure communication protocols, such as Transport Layer Security (TLS), are used by cloud service providers to protect data when it is being transferred between multiple cloud environments.

These technologies not only fulfill the GDPR requirements but also mitigate the risk of vendor lock-in. Vendor lock-in is when customers are so dependent on a cloud service provider that migrating to other cloud providers becomes almost impossible as stated by Armbrust et al. [4]. The obstacles that customers face are often significant expenses, restrictions from the law, or technical incompatibility. The findings of Opara-Martins et al. [12] highlight the significance of interoperability, portability, and adherence to standards in the realm of cloud computing.

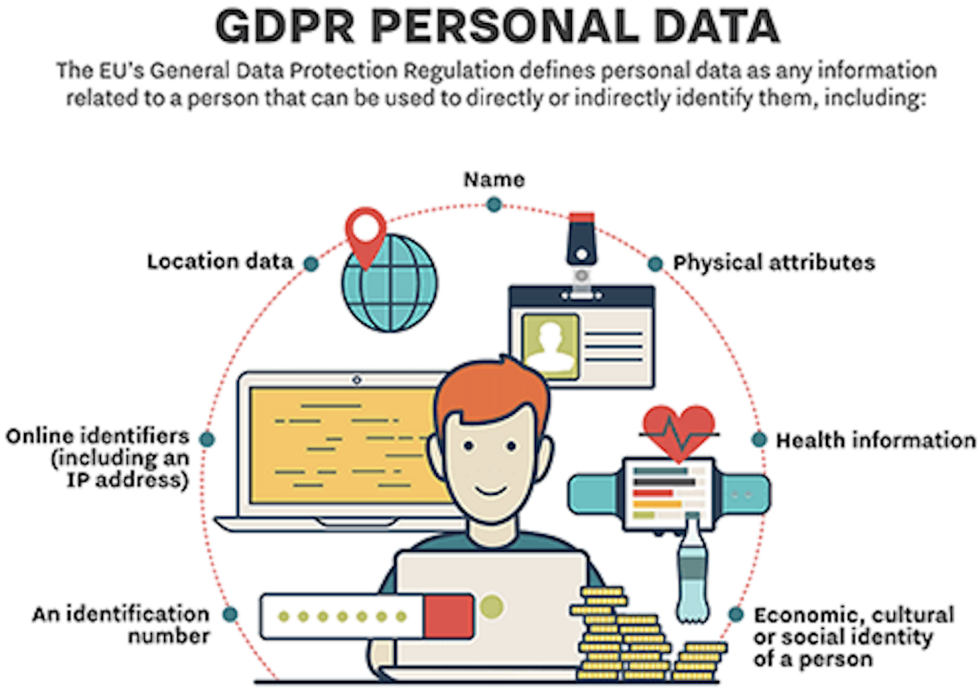


Figure 2: Personal Data defined by the GDPR.

2.3 Visibility regarding metadata and data minimization

According to Tolsma [21], the challenge of visibility and minimization of metadata encapsulate intended use and access rights, and the protection level afforded to metadata. In a nutshell, this boils down to data protection of metadata itself and access control as the mean of data protection. We have evaluated the data protection technologies with the CIA triad which stands for *confidentiality* (disclosure and restricted access to data), *integrity* (protecting data from improper alternation and destruction), and *availability* (ensuring reliable timely access and use of data) to check the compliance of the GDPR specific challenge, minimization and visibility of the data.

Data protection in terms of metadata

In general, Hassan et al. [8] stated that data privacy techniques can be divided into two main categories of non-cryptographic and cryptographic approaches. See Figure 3 for the diagram of an overview of data protection techniques.

The non-cryptographic techniques can be subdivided into variants of data anonymization, data splitting, and steganography. To begin with, data anonymization hides privacy information such as identifiers or attributes in the data by removing, covering, or adding noise to the data. Zhang et al. [26] introduced a historical probability-based noise generation strategy that injects noise to modify records based on the historical probability of noise requests. Although this idea significantly increased the availability compared to the traditional anonymization techniques by decreasing communication overhead and increasing the ease of computation, it is still vulnerable to unauthorized access to the data.

Second, data splitting divides sensitive data into different segments and randomly stores each fragment on different clouds. Therefore, even if a malicious user gets hold of a fragment of the data the whole data cannot be fully interpreted. Nonetheless, as a portion of the data can still be retrieved the privacy protection level is low.

Lastly, steganography is the practice to hide data within another message or object. Despite this practice creating low computation and offering a moderate level of protection, the data is not retainable compared to other techniques.

In contrast, cryptographic approaches are branched into diverse encryption methods. First, Yang et al. [25], Hassan et al. [8], Sun [19] advocated that in attribute-based encryption, the central main authority distributes generated private key to registered users and this key can be used to encrypt the data. This hinders unauthorized users, who do not have a matching set of ciphered attributes, from accessing the data and is effective when the registered number of users is very large. The drawback of this encryption is that it causes difficulty in updating the encryption file in the cloud.

An alternative encryption mechanism is searchable key encryption as discussed in Yang et al. [25], Hassan et al. [8], Sun [19]. In this encryption, the data is first encrypted locally before being uploaded to the cloud database and grants authorized users to perform query searches on encrypted data in the cloud database. In this way, users can avoid having to download and decrypt the data locally while providing a high level of protection.

The most dominant and promising approach was homomorphic encryption. In the studies, Takabi et al. [20], Yang et al. [25], Hassan et al. [8], homomorphic encryption displayed superiority in terms of protection and update issues that arose with attribute-based encryption. In homomorphic encryption, the data owner can encrypt the data by the homomorphic encryption algorithm and directly send the respective data to the cloud as it permits users to perform computations on encrypted data without decryption.

All in all, the proposed techniques in this subsection do indeed comply with the CIA with some shortcomings based on the property of the implementation. In short, non-cryptographic approaches depicted a lower level of security and data accuracy retainment due to the characteristic of modifying the metadata but provides lower overhead in computation compared to cryptographic methods. Conversely, cryptographic methods tend to create large overhead in computation and lead to lower availability but are suitable for a high level of protection while maintaining the data as it is.

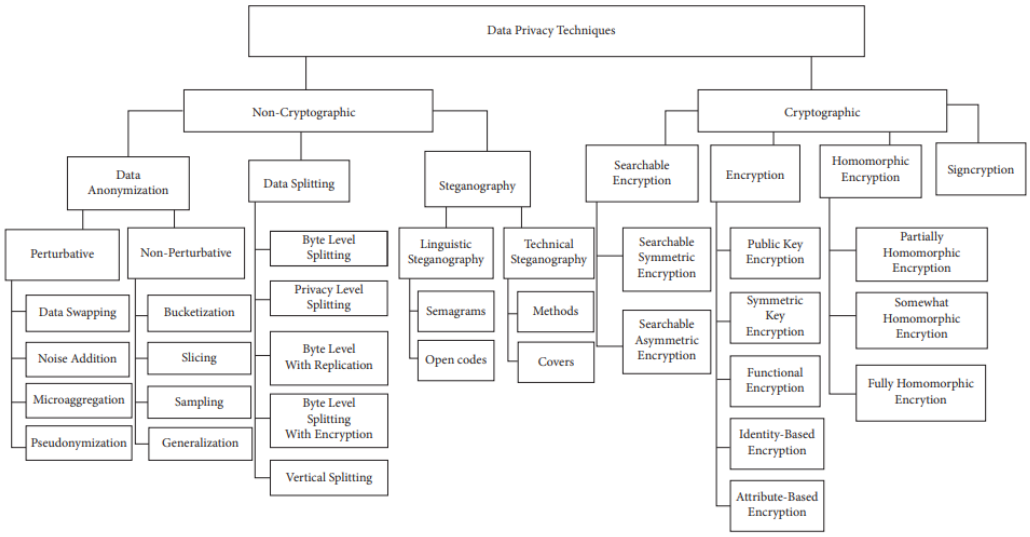


Figure 3: Overview of data protection techniques.

Data protection in terms of access control

Yang et al. [24] proposed a new blockchain-based access control framework in the cloud called AuthPrivacyChain. The main idea is to replace the traditional database-based access control with an immutable database, blockchain. This was achieved by first registering the cloud, data user, and metadata of the published resources and data owner to the blockchain via smart contracts. Next, whenever a user request for access to a resource the blockchain will intervene, and approval of

access will only be granted based on the permission record stored in the blockchain. See Figure 4 for a detailed system model of the framework. Experiments have shown that AuthPrivacyChain is capable of controlling the confidentiality and integrity of metadata with blockchain's characteristics of immutable blocks. In addition, they were able to mimic the throughput of the workloads of traditional database access control up to 400 smart contract transactions per second. Thus, the recommended approach can comply with all CIA triads with restrictions on availability. However, the study merely mentioned the issue in the context of big data in which workload throughput can be larger than 1000 workloads per second, and background on how to handle access updates of the permission in a blockchain-based access control system.

Davari and Bertino [6] disputed that blockchain-based access control in combination with an extension of XACML, attribute-based access control policy language, can offer data protection. The general idea of applying blockchain to permission control on access was similar to that of the previously discussed AuthPrivacyChain. The difference lies in extending existing XACML to adhere to the GDPR requirements. To accomplish this, they have included a policy that can describe and manage attributes of the resources, such as data subjects' identity, the purpose of data subjects' intention of sharing the data, validity time of the data, and resource type. In short, this approach also complies with CIA triads but has not been proven to be scalable in situations with large users and data.

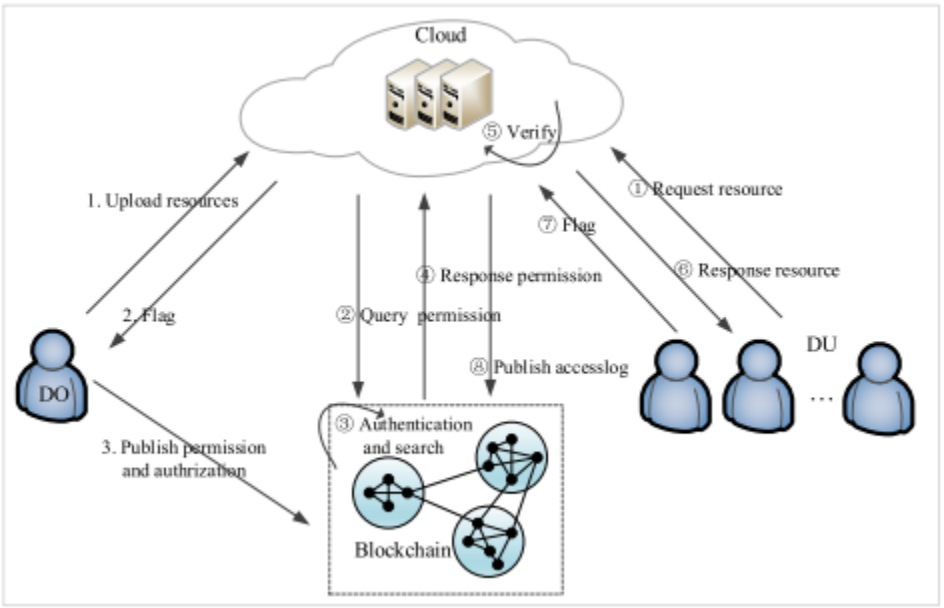


Figure 4: Blockchain-based access control architecture.

3 Discussion and Conclusion

To conclude, our review concentrated on how current or developing research in terms of data protection can address three GDPR-specific challenges: effective implementation of retention in the cloud, data portability, and visibility & minimization of metadata, proposed by the previous study, Tolsma [21]. The outcome depicted that all three challenges can be handled with existing and emerging practices with some weaknesses.

To address the challenge regarding implementing data retention effectively in the cloud, various approaches and technologies are being adopted, such as data classification and tagging, encryption and tokenization, data lifecycle management, geographically distributed data storage, and transparent data deletion processes. These approaches aim to improve data management, security, and privacy while meeting GDPR requirements. Implementing effective data retention strategies in the cloud, however, can be complex and challenging, especially when dealing with multi-jurisdictional requirements and managing backups. Organizations may require specialized knowledge and resources to navigate and comply with GDPR's privacy and data protection requirements.

In relation to the challenge of visibility and minimization of metadata, a number of researchers asserted visibility and minimization of the metadata itself by applying data anonymization, data splitting, steganography, and various encryptions. In brief, non-cryptographic practices depicted higher availability, but lower confidentiality and integrity in the CIA triad while cryptographic approaches demonstrated completely opposite behavior. This is because cryptographic approaches conceal the metadata and create more overhead during encryption or computing with the encrypted data. On the other hand, non-cryptographic techniques do not fully obscure the metadata, hence leading to different consequences. The remaining studies were concerned with promoting blockchain-based access control of the data. These approaches have been proven to be effective in terms of restricting unauthorized access to the data. However, the experiments were solely concentrated on data protection, and neither a proof of scalability in the context of big data nor a description of how to deal with the access permission update of this immutable access control system was provided.

Various strategies and technologies are being used to solve the problem of data portability for the controller. The objective of implementing these methods is to enrich the data portability experience for the customers and for the cloud providers (data controllers) themselves. Interoperability standards have a weakness and that is that a great number of participants is needed for it to work properly, but once it is widely used it can offer seamless data transfer. The use of APIs means that developers need to stay up to date with all the changes that can take place with an API, but it can provide a standardized way of communicating with the data controllers thus resulting in a more efficient data portability process. Containerization and virtualization can have extra performance overhead however encapsulating applications and their dependencies comes with benefits such as the consistent way of migrating from one cloud environment to another cloud environment.

On the basis of this literature review, further researches need to be conducted as there is no one-size-fits-all approach to solve the proposed challenges. Moreover, exploration is needed to come up with a uniform metric to evaluate all GDPR compliance in cloud computing. Another possible future research could be to evaluate the existing data protection technologies on, the upcoming, AI Act.

References

[1] S. Abiteboul and J. Stoyanovich. Transparency, fairness, data protection, neutrality: Data management challenges in the face of new regulation. *Journal of Data and Information Quality (JDIQ)*, 11(3):1–9, 2019.

[2] A. Z. Abualkishik, A. A. Alwan, and Y. Gulzar. Disaster recovery in cloud computing systems: An overview. *International Journal of Advanced Computer Science and Applications*, 11(9), 2020.

[3] Arcserve. 7 most infamous cloud security breaches. URL `https://www.arcserve.com/blog/7-most-infamous-cloud-security-breaches`.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4): 50–58, 2010.

[5] P. Bonatti, S. Kirrane, A. Polleres, and R. Wenning. Transparent personal data processing: The road ahead. In *Computer Safety, Reliability, and Security: SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, September 12, 2017, Proceedings 36*, pages 337–349. Springer, 2017.

[6] M. Davari and E. Bertino. Access control model extensions to support data privacy protection based on gdpr. *2019 IEEE International Conference on Big Data (Big Data)*, 2019.

[7] P. De Hert, V. Papakonstantinou, G. Malgieri, L. Beslay, and I. Sanchez. The right to data portability in the gdpr: Towards user-centric interoperability of digital services. *Computer law & security review*, 34(2):193–203, 2018.

[8] J. Hassan, D. Shehzad, U. Habib, M. U. Aftab, M. Ahmad, R. Kuleev, and M. Mazzara. The rise of cloud computing: data protection, privacy, and open research challenges—a systematic literature review (slr). *Computational Intelligence and Neuroscience 2022*, 2022.

[9] J. Holzel. Differential privacy and the gdpr. *Eur. Data Prot. L. Rev.*, 5:184, 2019.

[10] M. Kuperberg. Towards enabling deletion in append-only blockchains to support data growth management and gdpr compliance. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 393–400. IEEE, 2020.

[11] N. Kushwaha, P. Roguski, and B. W. Watson. Up in the air: Ensuring government data sovereignty in the cloud. In *2020 12th International Conference on Cyber Conflict (CyCon)*, volume 1300, pages 43–61. IEEE, 2020.

[12] J. Opara-Martins, R. Sahandi, and F. Tian. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5:1–18, 2016.

[13] C. Pahl. Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31, 2015.

[14] D. Petcu. Portability and interoperability between clouds: challenges and case study. In *Towards a Service-Based Internet: 4th European Conference, ServiceWave 2011, Poznan, Poland, October 26-28, 2011. Proceedings 4*, pages 62–74. Springer, 2011.

[15] E. Politou, E. Alepis, and C. Patsakis. Forgetting personal data and revoking consent under the gdpr: Challenges and proposed solutions. *Journal of cybersecurity*, 4(1):tyy001, 2018.

[16] E. Politou, A. Michota, E. Alepis, M. Pocs, and C. Patsakis. Backups and the right to be forgotten in the gdpr: An uneasy relationship. *Computer Law & Security Review*, 34(6): 1247–1257, 2018. ISSN 0267-3649. doi: https://doi.org/10.1016/j.clsr.2018.08.006. URL `https://www.sciencedirect.com/science/article/pii/S0267364918301389`.

[17] K. Rahul and R. K. Banyal. Data life cycle management in big data analytics. *Procedia Computer Science*, 173:364–371, 2020.

[18] M. Rhahla, S. Allegue, and T. Abdellatif. Guidelines for gdpr compliance in big data systems. *Journal of Information Security and Applications*, 61:102896, 2021.

[19] P. Sun. Security and privacy protection in cloud computing: Discussions and challenges. *Journal of Network and Computer Applications*, 160:102642, 2020.

[20] H. Takabi, J. B. Joshi, and G.-J. Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8.6:24–31, 2010.

[21] A. Tolsma. Gdpr and the impact on cloud computing. URL `https://www2.deloitte.com/nl/nl/pages/risk/articles/cyber-security-privacy-gdpr-update-the-impact-on-cloud-computing.html`.

[22] M. Vogt, A. Stiemer, and H. Schuldt. Polypheny-db: towards a distributed and self-adaptive polystore. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3364–3373. IEEE, 2018.

[23] Y. Wang and A. Shah. Supporting data portability in the cloud under the gdpr, 2018.

[24] C. Yang, L. Tan, B. X. Na Shi, Y. Cao, and K. Yu. Authprivacychain: A blockchain-based access control framework with privacy protection in cloud. *IEEE*, 8:70604–70615, 2020.

[25] P. Yang, N. Xiong, and J. Ren. Data security and privacy protection for cloud storage: A survey. *IEEE Access*, 8:131723–131740, 2020.

[26] G. Zhang, Y. Yang, and J. Chen. A historical probability based noise generation strategy for privacy protection in cloud computing. *Journal of Computer and System Sciences*, 78.5: 1374–1381, 2012.

[27] A. C. Zhou, Y. Xiao, Y. Gong, B. He, J. Zhai, and R. Mao. Privacy regulation aware process mapping in geo-distributed cloud data centers. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1872–1888, 2019. doi: 10.1109/TPDS.2019.2896894.

How are DevSecOps, ISO compliance, Security Audits, and AWS GovCloud contributing to cloud security practices?

David Freina
Department of Computer Science
University of Amsterdam
14181789
david.freina@student.uva.nl

Sushmita Thakur
Department of Computer Science
University of Amsterdam
14113430
sushmita.thakur@student.uva.nl

Jonas Wagner
Department of Computer Science
University of Amsterdam
14491818
jonas.wagner2@student.uva.nl

Abstract

This literature survey explores the unique challenges of cloud security and presents emerging approaches specific to different cloud computing domains. It analyses the topics of DevSecOps practices, ISO compliance, and dynamic auditing in the context of cloud computing. The survey examines the challenges and solutions related to achieving ISO compliance, integrating security into the software development lifecycle through DevSecOps, and addressing the unique challenges involved with cloud security audits and the need for dynamic auditing in cloud environments. The article concludes by examining AWS GovCloud as a practical solution developed to assess security audits, security-relevant features, and other services in detail. Additionally, this survey provides valuable insights into how cloud security, compliance, and auditing, contribute to enhancing trust and ensuring the security of cloud services.

Keywords: Cloud computing, Security, DevSecOps, ISO compliance, Auditing, AWS GovCloud.

1 Introduction

Cloud computing has revolutionized the way businesses and individuals store, access, and manage their data and applications. It offers a flexible and scalable solution by leveraging shared resources and virtualization technology. In this digital era, where data is generated at an unprecedented rate, cloud computing has become an integral part of our daily lives.

The concept of cloud computing dates back to the 1960s when the idea of resource-sharing and utility computing emerged. However, until the 2000s cloud computing didn't gain significant traction with the advancements in internet technology and virtualization. Tech giants like Amazon, Google, and Microsoft pioneered cloud services, facilitating infrastructure, platforms, and software over the Internet. Cloud computing offers a wide range of services that cater to diverse needs and requirements which includes on-demand access to computing resources, including storage, processing power, and software, without the need for on-premise physical infrastructure or hardware. This allows small or

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

large businesses to easily adjust and scale their resource allocation on a pay-as-you-go basis, ensuring cost efficiency and flexibility. Here are some examples of cloud computing services:

- **Infrastructure as a Service (IaaS):** With IaaS, users can access virtualized computing resources (servers and storage), while the provider manages the underlying infrastructure. Examples - AWS EC2, Microsoft Azure Virtual Machines, etc.
- **Platform as a Service (PaaS):** Developers can build, deploy, and manage applications using the provided platform without worrying about infrastructure. Examples - Google App Engine, Microsoft Azure App Service, etc.
- **Software as a Service (SaaS):** Users can access software applications over the internet without installation or maintenance. Examples - Salesforce and Microsoft Office 365 etc.
- **Database as a Service (DBaaS):** Users can manage database services without managing the infrastructure. Examples - Amazon RDS and Google Cloud SQL etc.
- **Function as a Service (FaaS):** Developers can deploy and run code without managing servers. Examples - AWS Lambda and Azure Functions etc.

These are just a few examples of the services offered by cloud computing that provide flexibility for users, enabling them to focus on their core tasks while relying on the cloud for computing resources. While cloud computing brings numerous advantages, it also presents challenges for users because of the complex layers embedded in the foundational framework. The top challenges for organizations moving their IT applications to the cloud services are - Security concerns, data privacy, and compliance are among others. As data is stored on remote servers and transmitted over the internet, ensuring the security and availability of sensitive information becomes critical. This paper explores four key practices of cloud security, outlined as follows: embracing DevSecOps in software development practices, complying with relevant ISO standards, then understanding the importance of cloud auditing and the challenges involved, and finally briefly exploring AWS GovCloud which is a specialized cloud environment specifically designed for handling data of sensitive nature and is currently offered only in the United States.

2 DevSecOps

SecOps is an abbreviation in the tradition of the similar term *DevOps*. While SecOps represents "Security and Operations", DevOps means "Development and Operations". There is also a merged version of these terminologies, *DevSecOps*, which can be defined together with the other abbreviations:

- **DevOps** is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality. [10]
- **SecOps** is the automation of processes and systems that identify and facilitate the remediation of security vulnerabilities in operational infrastructure. [19]
- **DevSecOps** refers to the integration of security principles and practices in DevOps through increased communication, collaboration, and integration between the development and operations teams with the security team. [27]

The collaboration of these activities is visualized in Figure 1 and shows, that security is an equal important topic that must not be neglected in today's software development and operating. To narrow the scope of this paper we will not discuss SecDev and focus on DevOps, SecOps, and DevSecOps. There are different combinations of DevSecOps like SecDevOps or DevOpsSec in the academic literature. Depending on the order, the respective author(s) prioritises the fields. [24] In the following the term DevSecOps is treated without any priority amongst the different fields.
At first the initial DevOps lifecycle is shortly described in subsection 2.1. Then the concepts and ideas of SecOps are introduced in subsection 2.2 to build necessary basics for the combination, DevSecOps, which is discussed in subsection 2.3.

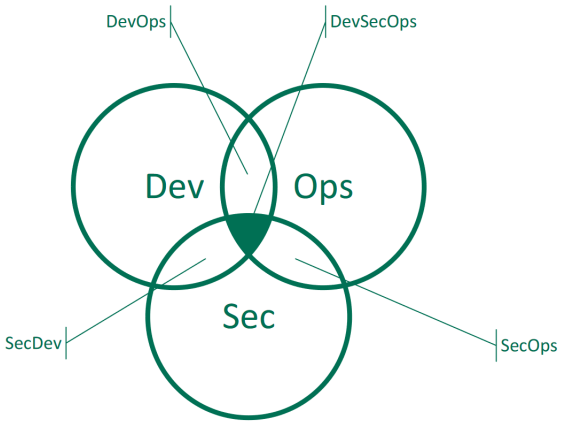


Figure 1: The fields of development, operations and security. [19]

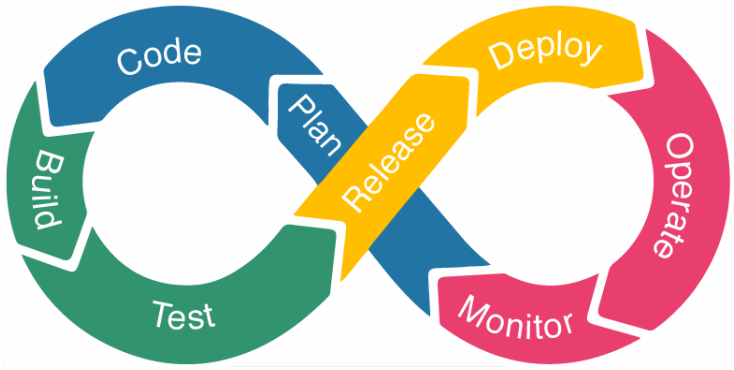


Figure 2: Different stages of the DevOps lifecycle [33]

2.1 DevOps

As defined above, the paradigm of DevOps aims to combine the development with actual deployment. The mentioned practices include multiple tools that allow developers, operators, project leads, and other involved specialists to conduct several phases as visualised in Figure 2:

- **Blue**: project planning on the basis of requirements and implementing the plan (e.g. agile development)
- **Green**: building and testing the application (e.g. build pipelines, integration tests)
- **Yellow**: publishing new versions of the application and bring them into production
- **Red**: operating and monitoring the application in a live environment with real data

Although the blue phase requires the most manual input, the other phases are configurable in a way that they are executed in an automated manner. The whole DevOps process is designed to run continuously to ensure a high quality application running in a stable environment. [12]
The process as described now is not explicitly taking care of security issues up to now.

2.2 SecOps

As illustrated in Figure 1, SecOps combines the fields of security and operations. To extend the scope of SecOps as defined above, there are several activities specifically related to security operations regarding to Jenkins and Steinke [19]:

- Understanding the security status of the IT operational infrastructure.
- Creating the guardrails of what can be deployed within the infrastructure.

- Aggregating security status of the infrastructure in real-time to enable timely issue resolution.
- Automating and managing all aspect of the infrastructure to enable self-healing.

These activities extend the right side of Figure 2 by introducing new security related aspects to it. What does that mean for the practical implementation of the infrastructure and the operation of the application, also on cloud environments?
Jenkins and Steinke name some examples for infrastructural configurations that include network, endpoint, certificate, and credential management, but also how to run the application (e.g. Kubernetes). On top of that they suggest automation and integration of the monitoring and incident handling which should trigger the system's recovery abilities. [19]
These are just some examples for better understanding of the abstract concept of SecOps, but the list of tools and activities for security related topics in the field of operations can be extended much further. But security also must be taken into account during other phases of the left half of Figure 2 rather than only development. To bring these concepts together, the term of DevSecOps aroused.

2.3 DevSecOps

DevSecOps is adding CAMS principles (culture, automation, measurement, sharing) to DevOps and extends even further with a special remark for security. Regarding to Myrbakken and Colomo-Palacios can be shortly summarized [23]:

- **Culture**: incorporating security aspects in each phase of the DevOps lifecycle (s. Figure 2) and aligning the team for a common goal of a secure development and operating application for the customer.
- **Automation**: security measurements and controls should be fully automated to keep up with the fast execution of DevOps. Furthermore, automated security must not impair the swiftness of DevOps.
- **Measurement**: extend the monitoring of the system's functionalities by adding measurements for potential security risks. The real-time operation benefits as well as the application's robustness against high load and potential threats.
- **Sharing**: knowledge and awareness about security issues should be shared across the involved people to achieve an overall improved DevSecOps lifecycle.

Applying these principles to the initial DevOps lifecycle in Figure 2 leads to a new model that is visualized in Figure 3. If an application should be developed in a cloud environment this newer model with additional security features along the phases of the the original DevOps lifecycle, is applicable. For each of the phases there is at least one security measurement that will improve specific aspects of the related phase but also the overall security of the application. [11] Following the continuous DevSecOps model and using the new feedback phase improves not only the process itself, but also helps with potential compliance and best practice issues due to new regulations or cloud environmental changes along the road.

3 ISO Compliance

ISO compliance refers to adhering to standards defined by the International Organization for Standardisation (ISO). This paper is detailing different ISO standards which are powerful guidelines to ensure proper information security in modern computer systems. The ISO worked together with the International Electrotechnical Commission (IEC) to create the following standards.

3.1 Standards

The following list provides an overview of the standards which are further explained in the following sections.

- ISO/IEC 27001: Information security management systems [14]
- ISO/IEC 27002: Information security, cybersecurity and privacy protection — Information security controls [15]

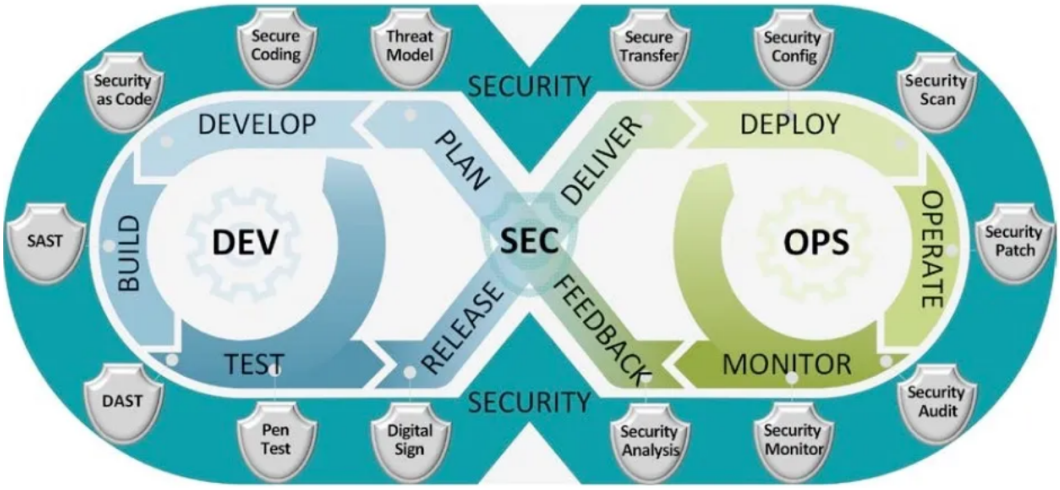


Figure 3: Applying SecOps to the DevOps lifecycle (reference model) [11]

- ISO/IEC 27017: Information technology — Security techniques — Code of practice for information security controls based on ISO/IEC 27002 for cloud services [16]
- ISO/IEC 27018: Information technology — Security techniques — Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors [17]
- ISO/IEC 27701: Security techniques — Extension to ISO/IEC 27001 and ISO/IEC 27002 for privacy information management — Requirements and guidelines [18]

The ISO/IEC 27K family of standards all incorporate a Plan-Do-Check-Act cycle (PDCA-cycle) to keep improving the controls they represent. In the planning phase, the first step is to define the various information assets and their corresponding security requirements. Next, information security risks are identified and evaluated, enabling organizations to develop appropriate controls and measures to mitigate these risks. Subsequently, the implementation of these controls and measures takes place. Finally, it is crucial to regularly and continuously monitor and review the performance of the system. [30]

3.1.1 ISO/IEC 27001 & ISO/IEC 27002 & ISO/IEC 27701

ISO/IEC 27001 and ISO/IEC 27002 are globally recognized standards that focus on information security management systems (ISMSes) and their respective implementation guidelines respectively. ISO/IEC 27001 provides a comprehensive framework for organizations to establish, implement, maintain, and continually improve their ISMS. The standard is applicable to companies of any size and sector, helping them effectively manage risks associated with the security of their data. By adhering to both standards, organizations can systematically identify and address security risks, protect their sensitive data, and enhance their overall information security. [14, 15]

With over 50,000 certified companies across more than 140 countries and various economic sectors, the ISO survey conducted in 2021 highlights the widespread adoption of ISO/IEC 27001. This significant number of certifications demonstrates the growing recognition and importance placed on implementing robust information security practices worldwide. [13]

In a survey conducted by Akinyemi et al. [9] in 2020, auditors, consultants, and researchers participated in a SWOT (strengths, weaknesses, opportunities, threats) analysis focused on the specification ISO/IEC 27001. The findings of the study revealed a generally positive perception among the participants regarding the identified 'Strengths' and 'Opportunities' associated with the specification.

ISO/IEC 27701 extends upon the standards and guidelines detailed in ISO/IEC 27001 and ISO/IEC 27002 and it specifically focuses on establishing, implementing, maintaining, and continually improving a Privacy Information Management System (PIMS) for privacy management within the scope of an existing ISMS of an organization. [18]

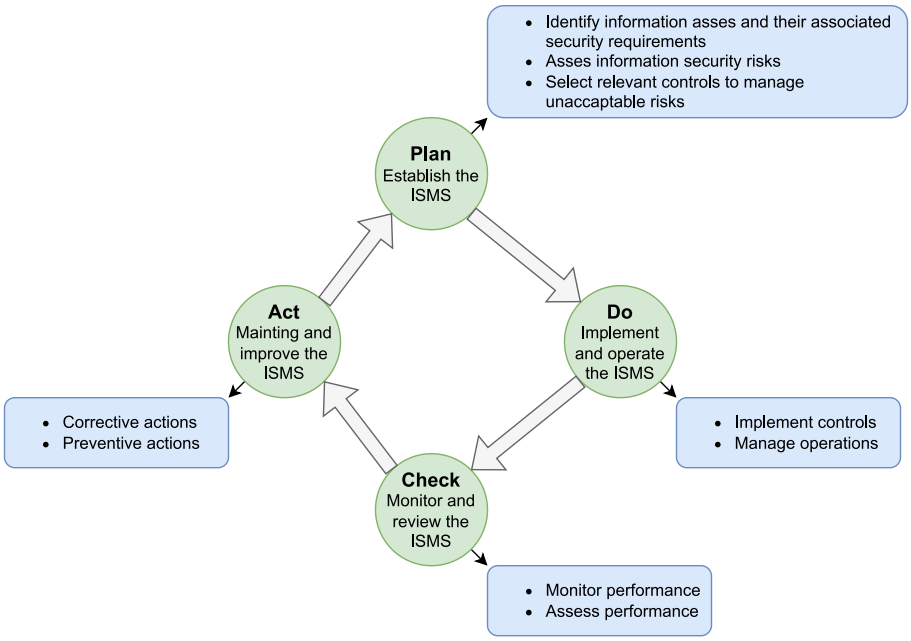


Figure 4: PDCA cycle ISO/IEC 27001 (Tissir et al. [30])

3.1.2 ISO/IEC 27017

ISO/IEC 27017 is a standard that provides guidelines for information security controls relevant to cloud services. It extends upon ISO/IEC 27002 by adjusting implementation guidelines for relevant topics with regard to cloud services and furthermore, it provides additional controls with implementation guidelines specifically for cloud services. The main aspect of this standard is to create a safer cloud service environment by reducing the risk of security-related problems.

There are seven additional controls added (compared to ISO/IEC 27002) which address the following [16]:

- Clarify roles and responsibilities when using cloud services: ISO/IEC 27017 helps to define clear responsibilities between the customer and the cloud service provider (CSP). This ensures accountability in the event of a security incident.
- Removal or return of assets at end of service: ISO/IEC 27017 addresses the proper handling of assets, e.g. data and systems, when terminating a contract with a CSP. It emphasizes the secure removal or return of assets to prevent unauthorized access.
- Isolated protection of virtual environments: ISO/IEC 27017 focuses on robust security measures to isolate virtual environments within a multi-tenant cloud environment. It ensures the separation of resources and prevents unauthorized access to sensitive data.
- Proper configuration of virtual machines: The standard provides guidelines for the secure configuration of virtual machines in cloud services. It takes considerations of virtualization technology and shared infrastructure into account in order to mitigate possible vulnerabilities.
- Cloud Service Management Operating Procedures: ISO/IEC 27017 addresses the management and control of administrative operations in the cloud environment (e.g. access control, user provisioning, privileged account management). This helps to prevent unauthorized actions and protect sensitive resources.
- Monitoring by cloud service users: ISO/IEC 27017 emphasizes enabling cloud customers to monitor and track activities related to their cloud services. This allows for increased visibility into the security of their assets.
- Coordinate virtual and physical network environments: ISO/IEC 27017 ensures the secure configuration of virtual and cloud network environments (e.g. network segmentation,

segregation, protection mechanisms). This helps to maintain data integrity and confidentiality within the cloud environment.

3.1.3 ISO/IEC 27018

ISO/IEC 27018 is standardized to use together with ISO/IEC 27002 in order to create a common set of security categories and controls for CSPs working with personally identifiable information (PII). The following objectives are outlined in the specification [17]:

- **Contracts**: Assist with the establishment of contracts between CSP customers and Personal Identifiable Information (PII) processors. It assists in defining the terms and conditions, rights, and responsibilities to ensure a clear understanding between the parties.
- **Accountability**: The standard helps CSPs fulfill their obligations as PII processors, whether through direct legal requirements or contractual agreements. It supports their compliance efforts, ensuring they take responsibility for protecting and managing PII appropriately.
- **Transparency**: It promotes transparency in PII processing. It assists PII processors in providing clear and comprehensive information to individuals whose data is being processed, enhancing trust and allowing individuals to make informed decisions about their data.
- **Audit & Compliance**: The standard enables customers to audit their data within complex multi-tenant and virtualized cloud environments.

3.2 Adoption

It is essential to distinguish between certification and compliance with an ISO standard. While obtaining an ISO certification can be beneficial for large companies in terms of gaining and maintaining customer trust (as shown in Table 1), the process can be resource-intensive in terms of time and cost, which may discourage some organizations from pursuing it as discovered by Mirtsch et al. [21].

As an alternative, compliance with an ISO standard offers a middle ground. Smaller companies can choose to comply with the requirements of a standard without necessarily seeking formal certification from the ISO. This approach allows them to adhere to the best practices outlined in the standard and demonstrate their commitment to meeting industry standards and customer expectations, without the administrative burden and financial implications associated with certification.

However, as mentioned in subsubsection 3.1.1 the ISO Survey 2021 [13] shows a widespread adaptation of ISO/IEC 27001 in the professional working field. Furthermore, Tariq. and Santarcangelo. [29] presented an overview of well-known public CSPs and their certification for different ISO standards. We have updated this table with the information currently available directly from the CSPs.

CSP	ISO/IEC 27001	ISO/IEC 27017	ISO/IEC 27018	ISO/IEC 27701
Amazon	✓	✓	✓	✓
Salesforce	✓	✓	✓	×
Microsoft	✓	✓	✓	✓
Google	✓	✓	✓	✓
IBM	✓	✓	✓	✓

Table 1: ISO certifications of major Cloud Service Providers (CSPs) (extended from Tariq. and Santarcangelo. [29])

4 Cloud Security Audit

IT audits aim to assess compliance with legal expectations for customer data protection and company standards for financial success amid security threats. Cloud security auditing involves assessing and evaluating the security controls and practices within a cloud environment to ensure secure and compliant operations. There are clear distinctions between cloud security and traditional IT security auditing. Moreover, Cloud security auditors require a combination of technical knowledge, familiarity with cloud terminologies, security-related industry expertise, and local regulatory understanding [28]. By thoroughly examining and identifying potential security risks, vulnerabilities, and compliance gaps, auditors assist cloud service providers (CSPs) in enhancing their platforms.

4.1 Challenges

In their survey on security mechanisms of prominent cloud service providers, Deepak et al. (2014) [26] highlight that cloud computing introduces distinct threats and risks that necessitate unique strategies for mitigating the risks, distinguishing it from traditional computing models. They categorize key challenges in cloud security, including data security, data loss, account or service traffic hijacking, and network security. Countermeasures involve implementing security measures at various levels, such as the kernel, storage, transmission, and access. These findings emphasize the complexity involved in cloud security auditing.

Here we also present some key challenges faced in Cloud Security Audits, as outlined by Ryoo et al. [28]:

Challenge	**IT security auditing practice**	**Cloud-specific challenge**
Transparency	Data and information security management systems are more accessible.	Data and security are managed by a third party.
Encryption	The data owner has control.	CSPs might be responsible for encryption.
Colocation	This rarely occurs.	CSPs heavily depend on this
Scale, scope, and complexity	These are relatively less	Auditors must be knowledgeable and aware of these differences.

Table 2: Cloud-specific Audit challenges [28]

- **Transparency:** Cloud customers lack visibility into how their data is processed and stored, hindering trust and increasing security risks.
- **Encyption (Safe-keeping of data at rest and in transit):** The physical distance between users and cloud service providers introduces a risk wherein third parties may gain unauthorized access to user data, compromising privacy. Encryption is vital for data security but in order to decrypt the information before performing calculations, users are required to provide the secret key to the server. Consequently, traditional cryptographic schemes are inadequate for processing data in the cloud [22].
- **Colocation (Shared Infrastructure):** Resource-sharing is a key factor driving the popularity of cloud services. However, cloud service providers (CSPs) must prioritize the effective management of administrator access permissions and the protection of user data. Wang et al. [25] discuss in their research on government cloud computing, the current situation of Government Cloud Computing in China which highlights the issues with cloud resource sharing among different government departments due to the conflict of interest for the data of sensitive nature.
- **Scale, Scope, and Complexity:** Auditing in cloud computing are fairly complex due to the sheer size of IT elements, rapidly changing technologies, and varying legal regulations across different countries' data centers.

4.2 Potential Solutions and Emerging Approaches

4.2.1 Standardization Frameworks

Building trust among potential customers is a crucial factor in facilitating the worldwide acceptance of cloud computing. The ISO 27000 series is a widely adopted IT security auditing standard [**?**] which has been discussed in the previous subsubsection 3.1.1. While considerable efforts are being made to develop preventive controls for security and privacy in the cloud, Ryan et al. [20] argue there is a notable gap when it comes to focusing on detective controls that enhance cloud accountability for potential risks and auditability.

4.2.2 Auditability

Auditability refers to the ease of auditing a system or environment. Poor audibility implies inadequate or nonexistent records and systems for efficient auditing of cloud processes [20]. In fact, traditional remote integrity-checking methods are limited to static data and cannot be effectively applied to auditing services in dynamic cloud environments where data is constantly updated. A **dynamic auditing protocol** improves cloud auditing by addressing security issues, and ensuring thorough auditing while reducing computing costs. The protocol supports confidentiality, dynamic data updates, and batch auditing across multiple clouds and owners. [32]

To address the problems of auditing, researchers have also developed a framework called **TrustCloud framework** [7], which focuses on detective controls to enhance accountability and simplify cloud security through five abstraction layers (workflow, data, system, laws & regulations, policies). It provides accountable cloud logs, regardless of the virtual or physical environment. As highlighted by Ryan et al. while the five layers may seem simple at first, the complexity lies in the different sub-components within each layer for various contexts. This framework enables file-centric logging, data-centric logging, and auditing in software services, protecting against both external and internal risks[20] by providing automated controls and policies which are invaluable tools that streamline the adoption of frameworks such as SOC 2 (Service Organization Control), ISO 27001, General Data Protection Law (GDPR), Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) etc. [7].

4.2.3 Homomorphic Encryption

In a survey on Homomorphic Encryption Schemes, Acar et al. [8] discuss that Legacy encryption systems pose significant privacy concerns as they rely on shared keys (public or private) for encrypted message exchange. This approach grants exclusive control over the data to users or service providers with the key, compromising privacy, particularly in popular cloud services. Moreover, even when keys are not shared, encrypted material remains vulnerable to unnecessary exposure to third parties. These issues can be addressed with, a specialized encryption scheme Homomorphic Encryption [1], which offers a promising solution by enabling third parties to perform operations on encrypted data without the need for prior decryption. This approach also eliminates the additional computation cost associated with traditional encryption methods. As another similar solution, in their study, Cong et al. [31] discusses a cloud auditing system that leverages HLA-based technology for data storage and implements a privacy-preserving public audit method. By integrating homomorphic linear authentication, the system ensures data privacy by enabling authentication for data block combinations. This safeguard prevents third-party auditors from accessing or viewing customer data, preserving data privacy.

5 AWS GovCloud

AWS (Amazon Web Services) is a renowned cloud service provider that offers a decentralized IT infrastructure. It was established in 2006 to extend the benefits of Amazon's extensive experience in managing large-scale IT infrastructure to other organizations. It offers various services like EC2 and S3, each serving different purposes and utilizing different databases such as RDS, DynamoDB, and Elastic Cache. **GovCloud** is a secure cloud solution designed specifically for state and federal government customers.

AWS GovCloud is a combination of Amazon Web Services (AWS) and GovCloud [2] which offers different data classification levels, from unclassified to top secret, to meet government regulations and ensure compliance. It provides a comprehensive set of features to address various requirements [2]:

- **Safeguard sensitive data:** Server-side encryption is implemented within Amazon S3 to safeguard the data at rest. Furthermore, the security keys associated with the encrypted data are managed securely through AWS CloudHSM or AWS Key Management Service (AWS KMS).
- **Strengthen identity management:** These measures include restricting access to sensitive data based on individual credentials, as well as considering factors such as time and location. Identity federation, easy key rotation, and other access control tools also ensure comprehensive access control within the system.

- **Improve cloud visibility:** Users get visibility into access and usage of sensitive data with AWS CloudTrail. This logging service monitors API activity and is operated by U.S. citizens for added security.
- **Protect accounts and workloads:** Wrokloads and accounts are protected with continuous security monitoring provided by Amazon GuardDuty.

Furthermore, AWS GovCloud is designed to facilitate the management of regulated data by implementing two key components [3]:

1. Restricting physical and logical administrative access exclusively to AWS personnel who are U.S. citizens. This stringent measure ensures that only authorized individuals with appropriate clearances can access and manage the system.
2. Offering FIPS 140-2 endpoints, which adhere to the Federal Information Processing Standard (FIPS) 140-2 for cryptographic modules [5]. These endpoints provide a secure environment for handling sensitive data, ensuring compliance with rigorous encryption standards.

It's important to note that while AWS does provide cloud services to customers in various countries worldwide through its standard AWS regions which adhere to local regulations and compliance requirements, AWS GovCloud is currently limited to the United States. CSPs face limitations in deploying private cloud services on a global scale, especially in government-oriented businesses because CSPs must comply with local laws which differ geographically for eg. General Data Protection Regulation (GDPR) [6] and Digital Services Act (DSA) [4] are the set of legal frameworks that regulate digital services in the EU area.

6 Discussion and Conclusion

In summary, cloud security is a critical aspect of modern computing, we have explored various aspects of cloud security practices, beginning with some background of cloud computing and the prevalent security challenges encountered in today's cloud landscape. It is reasonable to assert that adopting practices such as SecOps, DevSecOps, cloud auditing, complying with ISO standards, and deploying specialized cloud environments like AWS GovCloud contribute to building a secure cloud environment.

The implementation of DevSecOps in the cloud has been explained, highlighting its workflow and essential characteristics for effective integration. By practicing the DevSecOps-based development framework, organizations can cultivate a security-centric culture, emphasizing the use of security measures to safeguard systems and assess security risks. Furthermore, we have presented ISO standards pertaining to Information Security Management Systems (ISMS) and cloud security. These standards not only serve as certification references but also stimulate the progress of cloud services by offering guidance on constructing protected systems. We have discussed the definition of cloud audits and the challenges they entail. These audits serve as indispensable tools for assessing the safety of cloud systems. In cloud security, audits assume a critical role by facilitating the assessment of security controls and practices within cloud environments. While significant progress has been made in designing effective cloud security audit frameworks and standards, it is important to acknowledge that there is still a need for comprehensive standards that encompass all aspects of auditing cloud systems. Therefore, the journey toward attaining robust cloud security remains an ongoing endeavor. Moreover, we have presented a real-world example of a cloud environment - AWS GovCloud, to emphasize the importance of addressing security concerns in cloud environments. AWS GovCloud has been tailored to meet the specific requirements of government agencies and organizations handling sensitive workloads employing encryption techniques such as FPS 140-2 and FPS 140-4 which are the standard encryption requirements for data protection by all federal authorities within the US. This example underscores the growing attention given to cloud security and highlights the ongoing challenges in this field especially for government agencies.

As we notice that despite notable advancements in bolstering cloud security, challenges persist. The importance of cloud security continues to grow, demanding further efforts to address the remaining obstacles and ensure the utmost safety of cloud systems.

References

[1] https://en.wikipedia.org/wiki/Homomorphic_encryption. Accessed: 2023-05-26.

[2] https://aws.amazon.com/govcloud-us/?whats-new-ess.sort-by=item.additionalFields.postDateTime&whats-new-ess.sort-order=desc, . Accessed: 2023-05-26.

[3] https://docs.aws.amazon.com/govcloud-us/latest/UserGuide/whatis.html, . Accessed: 2023-05-26.

[4] Digital services act. https://digital-strategy.ec.europa.eu/en/policies/digital-services-act-package. Accessed: 2023-05-26.

[5] Federal information processing standard 140-2, security requirements for cryptographic modules. https://csrc.nist.gov/publications/detail/fips/140/2/final. Accessed: 2023-05-26.

[6] General data protection regulation. https://gdpr.eu/. Accessed: 2023-05-26.

[7] Trustcloud. https://www.trustcloud.ai/all-frameworks/. Accessed: 2023-05-26.

[8] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. 2018.

[9] Iretioluwa Akinyemi, Daniel Schatz, and Rabih Bashroush. Swot analysis of information security management system iso 27001. *International Journal of Services Operations and Informatics*, 10(4):305–329, 2020. doi: 10.1504/IJSOI.2020.111297.

[10] Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A Software Architect's Perspective*. SEI Series in Software Engineering. Addison-Wesley, New York, May 2015. ISBN 978-0-13-404984-7. URL http://my.safaribooksonline.com/9780134049847.

[11] Chief Information Officer Department of Defense. Dod enterprise devsecops reference design. August 2019. URL https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf.

[12] Mayank Gokarna. Devops phases across software development lifecycle. January 2021. doi: 10.36227/techrxiv.13207796.v2.

[13] ISO Survey 2021. ISO Survey of certifications – 2021. Survey, International Organization for Standardization, Geneva, CH, September 2022.

[14] ISO/IEC 27001:2022. Information security management systems. Standard, International Organization for Standardization, Geneva, CH, October 2022.

[15] ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection – Information security controls. Standard, International Organization for Standardization, Geneva, CH, March 2022.

[16] ISO/IEC 27017:2015. Information technology – Security techniques – Code of practice for information security controls based on ISO/IEC 27002 for cloud services. Standard, International Organization for Standardization, Geneva, CH, December 2015.

[17] ISO/IEC 27018:2019. Information technology – Security techniques – Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors. Standard, International Organization for Standardization, Geneva, CH, January 2019.

[18] ISO/IEC 27701:2019. Security techniques – Extension to ISO/IEC 27001 and ISO/IEC 27002 for privacy information management – Requirements and guidelines. Standard, International Organization for Standardization, Geneva, CH, August 2019.

[19] Jim Jenkins and Gerhard Steinke. What should your devsecops program look like?, October 2021. URL https://www.researchgate.net/publication/356262132_What_Should_Your_DevSecOps_Program_Look_Like.

[20] Ryan Ko, Peter Jagadpramana, Miranda Mowbray, Siani Pearson, Markus Kirchberg, Qianhui Liang, and Bu Lee. Trustcloud: A framework for accountability and trust in cloud computing. pages 584–588, July 2011. doi: 10.1109/SERVICES.2011.91.

[21] Mona Mirtsch, Knut Blind, Claudia Koch, and Gabriele Dudek. Information security management in ict and non-ict sector companies: A preventive innovation perspective. *Computers Security*, 109:102383, 2021. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2021.102383. URL `https://www.sciencedirect.com/science/article/pii/S0167404821002078`.

[22] Mohammed Mohammed and Fadhil Abed. An improved fully homomorphic encryption model based on n-primes. *Kurdistan Journal of Applied Research*, 4:40–49, 10 2019. doi: 10.24017/science.2019.2.4.

[23] Håvard Myrbakken and Ricardo Colomo-Palacios. Devsecops: A multivocal literature review. pages 17–29, September 2017. ISBN 978-3-319-67382-0. doi: 10.1007/978-3-319-67383-7_2.

[24] Rennie Naidoo and Nicolaas Möller. Building software applications securely with devsecops: A socio-technical perspective. *European Conference on Cyber Warfare and Security*, 21:198–205, June 2022. doi: 10.34190/eccws.21.1.295.

[25] Wang Ning, Xie Xiaoshan, Li Hui, Wang Xuehua, and Qin Xuezhi. Survey of application and research on government cloud computing in china. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pages 140–143, 2015. doi: 10.1109/WI-IAT.2015.194.

[26] Deepak Panth, Dhananjay Mehta, and Rituparna Shelgaonkar. A survey on security mechanisms of leading cloud service providers. *International Journal of Computer Applications*, 98(1): 34–37, 2014.

[27] Roshan Rajapakse, Mansooreh Zahedi, Muhammad Ali Babar, and Haifeng Shen. Challenges and solutions when adopting devsecops: A systematic review. *Information and Software Technology*, 141:106700, August 2021. doi: 10.1016/j.infsof.2021.106700.

[28] Jungwoo Ryoo, Syed Rizvi, William Aiken, and John Kissell. Cloud security auditing: Challenges and emerging approaches. *IEEE Security Privacy*, 12(6):68–74, 2014. doi: 10.1109/MSP.2013.132.

[29] Muhammad Imran Tariq. and Vito Santarcangelo. Analysis of iso 27001:2013 controls effectiveness for cloud computing. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP*, pages 201–208. INSTICC, SciTePress, 2016. ISBN 978-989-758-167-0. doi: 10.5220/0005648702010208.

[30] Najat Tissir, Said El Kafhali, and Noureddine Aboutabit. Cybersecurity management in cloud computing: semantic literature review and conceptual framework proposal. *Journal of Reliable Intelligent Environments*, 06 2021. doi: 10.1007/s40860-020-00115-0.

[31] Cong Wang, Sherman S.M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers*, 2013. doi: 10.1109/TC.2011.245.

[32] Kan Yang and Xiaohua Jia. An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 24(9):1717–1726, 2013. doi: 10.1109/TPDS.2012.278.

[33] Ravi Teja Yarlagadda. Devops and its practices. *SSRN Electronic Journal*, 9:2320–2882, March 2021.

Exploring iPaaS: An Comprehensive Analysis of Research Status, Solution Issues, and Future Research Directions

Yiming Wu
13646885
yiming.wu2@student.uva.nl

Bowen Liang
14730936
bowen.liang@student.uva.nl

Zhe Liu
14723670
zhe.liu2@student.uva.nl

Abstract

This report presents a literature study on Integration Platform as a Service (iPaaS), a solution that enables the integration of data, applications, and services. A comprehensive collection of related papers is gathered and classified to gain insights into the current research status of iPaaS. The report also discusses issues associated with current iPaaS providers, providing a distilled presentation on the limitations and challenges. Additionally, the study summarizes possible directions for future research, highlighting areas that require further investigation.

1 Introduction

Integration platform as a service (iPaaS) (13) is a platform that enables individuals, organizations, and enterprises to integrate and connect various applications, systems, and data sources through connectors. The primary purpose of iPaaS is to facilitate the seamless integration of disparate components using low-code or even no-code principles.

For example, let's consider an e-commerce platform integrated with a shipping and logistics system. These two components operate independently and store valuable data in different formats. By leveraging iPaaS, these components can be connected through connectors provided by the iPaaS provider. This integration enables the automatic generation of shipping labels, tracking numbers, and order status updates. When a customer places an order online, the shipping and logistics system is automatically notified, providing customers with timely shipping updates within the e-commerce platform.

In this report, we perform a literature study on iPaaS. Specifically, we aim to present the current status of iPaaS research, discuss the challenges associated with its application, and identify potential research directions for iPaaS.

The structure of this report is as follows. Section 2 provides background information on iPaaS, including its development, and presents an overview of selected iPaaS providers. In Section 3, we compare iPaaS with similar technologies. Section 4 defines our research questions. Section 5 outlines our methods for literature collection and analysis. Section 6 summarizes our findings and results pertaining to the proposed research questions. Section 7 discusses the advantages and disadvantages of our study. Finally, in Section 8, we conclude our report.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Figure 1: Complexity of implementing connectors when adding new compoments

2 Background

2.1 Enterprise integration

Enterprise integration, a topic that revolves around connecting different components and exchanging various data across multiple systems, has always been a challenge for the information technology departments of organizations.

One solution is Point-to-Point Integration, which involves building a custom connector between any pair of applications. The idea is simple; however, it does have drawbacks, such as the quadratic increase in complexity. For example, as illustrated in Figure 1, adding one component to a three-component system requires the implementation of three additional connectors while adding two components will necessitate the inclusion of seven extra connectors. In general, the total number of connectors needed in an n-component assembly can be calculated using the formula

$$total\ number = \frac{n * (n-1)}{2}.$$

An improved approach for system integration is Enterprise Application Integration (EAI) (19), often involving middleware technologies, including a broker that transforms and routes data, among other functionalities, to integrate all the components. However, while this design reduces coupling between separate applications, the broker becomes a single point of failure.

To decrease the workload of the single broker component, Enterprise Service Bus (ESB) (20) comes into play as an evolution of EAI. It uses the concept of a bus that is analogous to computer hardware. Therefore, unlike broker-based EAI, which hard-coded all the integration logic in the broker component, the bus in an ESB architecture now only handles message routing. All other necessary functionalities for system integration, such as security and transaction processing, are implemented as separate modules. The system remains loosely coupled, and new components can be easily added. This makes ESB a lightweight and service-oriented architecture for system integration capable of scaling.

2.2 The development of iPaaS

The emergence of cloud computing brought significant changes to information technology and enterprise landscapes. Organizations began migrating their systems, applications, and infrastructures to the cloud to leverage its scalability and flexibility, and to reduce the cost and efforts to maintain the underlying hardware. iPaaS emerged as a response to the need for cloud-centric integration solutions and to break silos. It provides a platform for integrating various applications, data sources, and services.

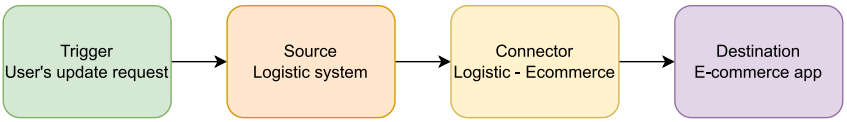


Figure 2: An integration of the e-commerce example

At present, there are a lot of iPaaS solution providers available, such as Workato [1], Informatica [2], Celigo [3], boomi [4], MuleSoft [5], and ROMA Connect [6]. A more complete list can be found in Gartner [7].

2.3 Technical details of iPaaS

Technically speaking, iPaaS is a suite of cloud services that enable customers and enterprises to define, develop, and manage integration flows between different components at a cloud scale. In practice, an iPaaS solution provider offers a unified and one-stop platform for enterprises to automate business processes. It has hundreds or even thousands of built-in connectors that bridge different pairs of applications and services together. Generally, an intuitive graphical user interface, along with a dashboard and some tools, is presented to the user. There is a developer workspace that is available to create integration flows. To assist non-technical users in building their pipelines, a low-code or even no-code drag-and-drop designer is provided within the platform. Some providers also supply users with pre-defined flows, providing them with an out-of-the-box experience. Furthermore, error management and processing functionality are also provided.

Different solutions may have unique sets of connectors, distinct data mapping technologies, or diverse higher-level functionalities. Each platform typically offers documentation and training courses.

Figure 2 shows an integration flow for the e-commerce and logistics system discussed in the previous section. As illustrated, the flow is activated when a user requests an update on the shipping status. The source data from the logistics system then flows to the destination e-commerce application through the corresponding connector. Mapping the tracking number to the order ID can be added. Additional filtering may also be performed to reduce the amount of data transferred.

3 Related work

In this section, we review the existing literature on EAI, ESB, and iPaaS. We explore the relationships between these approaches and provide a comparative analysis of their features and capabilities.

3.1 EAI and ESB

Soomro et al. (21) reviewed the development and evolution of EAI, starting from information-oriented approaches that involve the movement of data between different storage systems and processes as per requirements. They then explored interface-oriented approaches, which enable access to business processes and data through APIs. The authors further discussed process-oriented approaches, where one application can access the methods of another application. Finally, they highlighted service-oriented approaches, specifically the enterprise service bus (ESB). The authors acknowledged the usefulness of ESB but also identified challenges faced by EAI as an integration technology.

Goel (22) performed a comprehensive comparison between EAI and ESB, considering factors such as installation and administration efforts, costs, scalability, standardizations, and support for service-oriented architecture. The study found that ESB generally outperforms EAI in terms of costs, scalability, standardizations, and service-oriented architecture support.

[1] https://www.workato.com/products/ipaas
[2] https://www.informatica.com/nl/products/cloud-integration.html
[3] https://www.celigo.com/
[4] https://boomi.com/
[5] https://www.mulesoft.com/integration-solutions/api/ipaas
[6] https://www.huaweicloud.com/intl/en-us/product/roma.html
[7] https://www.gartner.com/

3.2 ESB and iPaaS

Both iPaaS and ESB, have been proposed and developed to address the enterprise integration challenge. The selection of an appropriate method and its corresponding solutions has always been an open research question. Conflicting conclusions exist regarding the choice of a suitable technology (8). Zhang et al. (8) conducted an analysis based on economic models and compared integration scenarios, complexity, scalability, and extension. The authors concluded that ESB is suitable for on-premises software, while iPaaS is suitable for cloud-based services. ESB also exhibits greater complexity compared to iPaaS. In terms of scalability, ESB and iPaaS both demonstrate scalability, but in different scenarios. Additionally, iPaaS provides better support for the Internet of Things (IoT) and Software as a Service (SaaS) integration.

4 Research questions

In this literature study, a collection of papers on iPaaS is analyzed, reviewed, and summarized, aiming to address the following research questions:

1. What is the current research status of iPaaS?
2. What are some issues with current iPaaS solutions?
3. What are some possible directions for future iPaaS research?

5 Method

5.1 Literature collection and selection

To collect iPaaS-related papers, we searched the Scopus database using the query string

TITLE-ABS-KEY ("iPaaS" OR "Integration Platform as a Service"),

following a similar approach as described in (13). Subsequently, we manually inspected each paper in the search results to filter out irrelevant papers. It is worth noting that some of these irrelevant papers were included in the search results due to the presence of other terminologies that share the acronym "iPaaS". An example of such terminology is Ileal Pouch-Anal Anastomosis.

5.2 Literature analysis and categorization

After collecting the papers, we categorized them using a similar approach as described in (13). We identified two distinct groups. The first group focused on discussing iPaaS from a technological perspective, exploring its technical aspects, functionalities, and implementation considerations. The second group primarily emphasized the business benefits of adopting iPaaS for enterprise applications, examining the service designs, and solution providers.

6 Results

6.1 Research question 1: What is the current research status of iPaaS?

Analysis of the timeline and the number of our collected papers indicates that iPaaS is a relatively new and evolving area. As shown in Figure 3, which represents the distribution of these papers over the years, there are only 18 papers in total, with the first well-known paper (1) published in 2012. Furthermore, a growing interest in iPaaS has been observed from 2020 onwards.

Following a similar method as in (13), we classified the collected papers into two groups: Business-intensive and Technological. The results are presented in Table 1, Figure 4, and Figure 5. Contrary to the conclusion in (13), which stated that the majority of papers are technology-focused, our findings indicate that the number of papers in each group is approximately evenly distributed, with 10 papers in the Business-intensive group and 8 papers in the Technological group. Since (13) was published in 2022, this observation implies an increasing interest in the business benefits of adopting iPaaS since that time, as also evident from Figure 4.

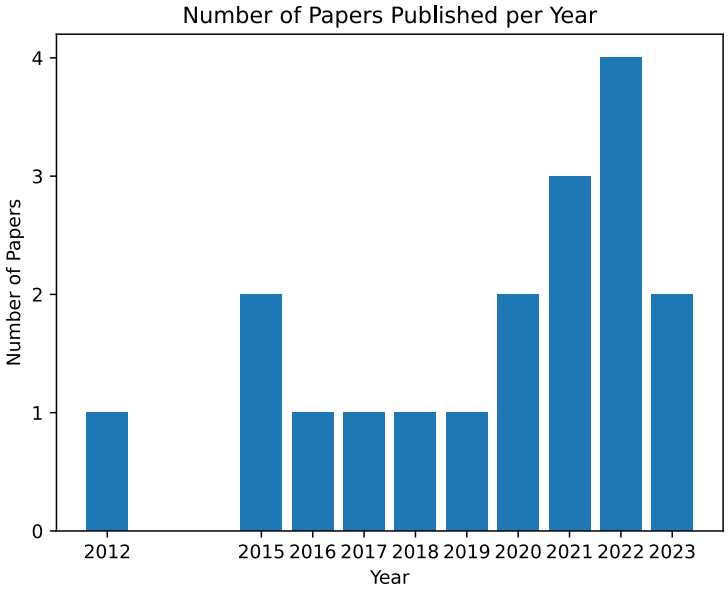


Figure 3: Number of papers published from 2012

Table 1: List of Authors, Year, and Type

Authors	**Year**	**Type**
Bolloju N. & Murugesan S.(1)	2012	Business-intensive
Manilal, S. & Theertha, V.S.(2)	2015	Technological
Jafarov, N., & Lewis, E.(3)	2015	Technological
Suzic, B.(4)	2016	Technological
Ebert, N. et al.(5)	2017	Business-intensive
Theilig, M. M. et al.(6)	2018	Business-intensive
Srimathi H. & Krishnamoorthy A.(7)	2019	Technological
Zhang, X., & Yue, W. T.(8)	2020	Business-intensive
Cestari, R. H. et al.(9)	2020	Technological
Neifer, T. et al.(10)	2021	Business-intensive
Frantz, R. Z. et al.(11)	2021	Technological
Hyrynsalmi, S. M. et al.(12)	2021	Business-intensive
Hyrynsalmi, S. M.(13)	2022	Technological
Sänger, N., & Abeck, S.(14)	2022	Business-intensive
Hyrynsalmi, S. M.(15)	2022	Technological
Hyrynsalmi, S.(16)	2022	Business-intensive
Hyrynsalmi, S., & Smolander, K.(17)	2023	Business-intensive
Huang, R.(18)	2023	Business-intensive

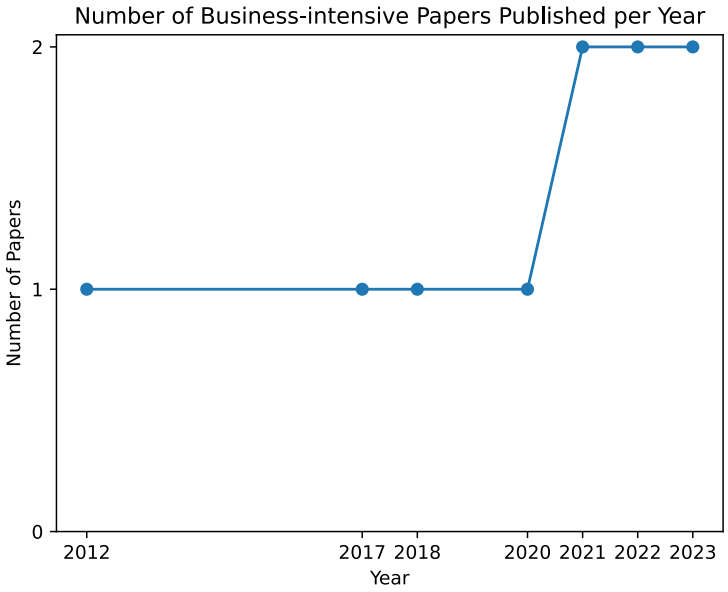


Figure 4: Number of Business-intensive papers published from 2012

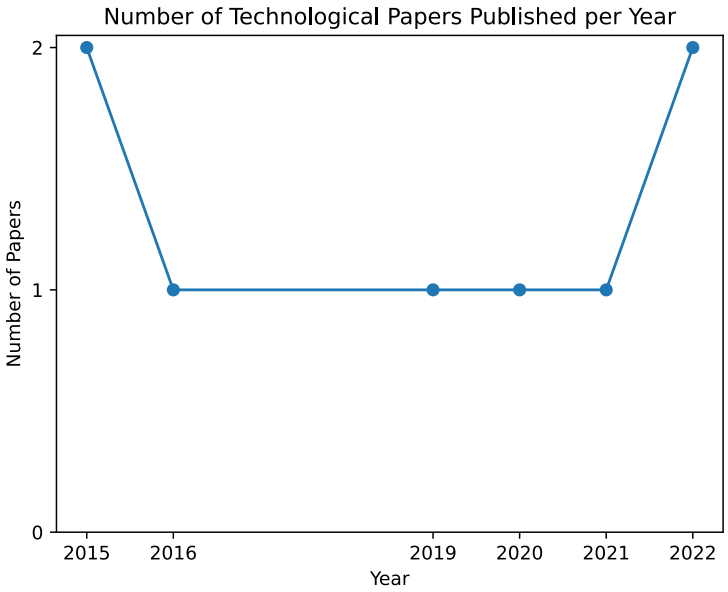


Figure 5: Number of Technological papers published from 2012

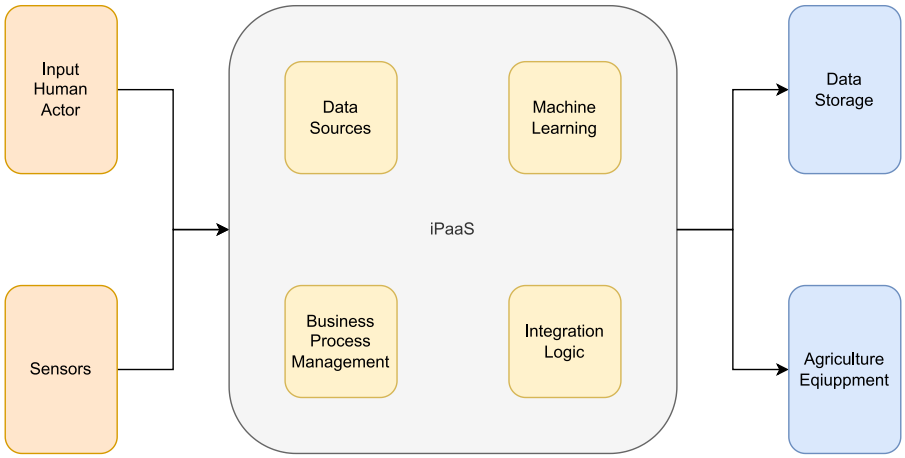


Figure 6: An overview of the design

6.2 Research question 2: What are some issues with current iPaaS solutions?

While iPaaS is a relatively new research field, there are over 60 solution providers listed in Gartner [8]. Additionally, practical applications of iPaaS technology in real-world scenarios, such as (9) and (18), have been observed. However, several papers, including (10), have pointed out issues with existing iPaaS solutions.

The first issue arises from the variety of business requirements (10). Although popular iPaaS solution providers offer thousands of connectors, they cannot cover all possible application scenarios. According to an interviewee in [2], there is no connector available that meets their specific requirement.

The second issue is related to the previously mentioned one. Customizing connectors to meet business demands lacks a cost-benefit ratio and can result in high costs (9). This issue is particularly significant for small and medium-sized enterprises with specific solutions.

The third issue is the lack of standardization for data models and data exchange (10). While iPaaS allows the integration of data from various sources, different components may produce data that differ in nature. For example, different sources may collect data with varying granularities: application A collects data on a daily basis, while application B collects data on a second-by-second basis.

There are also other practical issues, including data security, protection, and privacy, the possible failures of connectors, and the documentation and technical support provided by solution providers (10).

Furthermore, concerns about brand-locking have also been raised (9).

6.3 Research question 3: What are some possible directions for future iPaaS research?

As iPaaS is relatively new, there are many opportunities for future research. For example, addressing the previously mentioned issues, improving usability, and developing technologies to facilitate digital transformation. Since the primary goal of iPaaS is to integrate various systems, applications, and data sources without the need for licensed middleware or hardware, the application area can be extended beyond enterprises, including IoT and industrial control, among others.

Research cases that explore such possibilities have already emerged. For instance, (9) implemented a prototype iPaaS solution for grain storage and processing, as shown in 6. The system integrates heterogeneous components such as humidity and temperature controllers and environment monitors. One of the goals was to reduce the need for human interactions, leading to the integration of automated decision-making modules. The system has an open design, allowing for the integration of additional systems. Consequently, as the authors pointed out, it is possible to utilize the system in applications other than agricultural activities.

[8] https://www.gartner.com/reviews/market/integration-platform-as-a-service-worldwide

7 Discussion

This report aimed to gain insight into the current status of iPaaS research, issues with existing iPaaS solutions, and possible directions for future study.

To answer these questions, we collected, reviewed, and analyzed a comprehensive set of related papers. By examining the number of papers published per year and comparing our findings with (13), we found significant research progress in 2022. Firstly, it had the highest number of papers published. Secondly, more research projects started to focus on business-intensive topics.

Although our literature study is comprehensive, there are some limitations worth noting in our report.

First, the total number of publications is small, which means that the research trends may not fully reflect our predictions.

Second, iPaaS technology continues to evolve. As a result, the findings and conclusions in our report may quickly become outdated. In fact, the term iPaaS itself has been evolving, leading to a clearer and distinguishable definition and scope separate from ESB and EAI. Due to the broader definition of iPaaS, there is now a more enterprise-focused term called EiPaaS [9], which stands for Enterprise Integration Platform as a Service.

8 Conclusion

In this literature study, we collected a comprehensive set of publications about iPaaS from 2012 to the present to examine the current research status, summarize issues with solution providers, and identify future research opportunities. We concluded that the field is relatively new and the number of publications is limited. We listed common issues, including the variety of connectors, potential high costs, and lack of standardization, among others. Additionally, we highlighted possible research directions to improve existing iPaaS technology and explore new application areas, such as the Internet of Things.

References

[1] Bolloju, N., & Murugesan, S. (2012, August). Cloud-based B2B systems integration for small-and-medium-sized enterprises. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (pp. 477-480).

[2] Manilal, S. & Theertha, V.S.. (2015). Preventing cloud integration challenges using iPaaS. 10. 1972-1976.

[3] Jafarov, N., & Lewis, E. (2015, July). Reinterpreting the principles of SOA through the cybernetic concepts of VSM to design the ESB as iPaaS in the cloud. In 2015 Science and Information Conference (SAI) (pp. 850-858). IEEE.

[4] Suzic, B. (2016, April). User-centered security management of API-based data integration workflows. In NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium (pp. 1233-1238). IEEE.

[5] Ebert, N., Weber, K., & Koruna, S. (2017). Integration platform as a service. Business & Information Systems Engineering, 59, 375-379.

[6] Theilig, M. M., Pröhl, T., & Zarnekow, R. (2018). Requirements Analysis for an Open iPaaS: Exploring the CSP, ISP, and SME View.

[7] Srimathi, H., & Krishnamoorthy, A. (2019). Integration Of Student System Using iPaaS. International Journal of Scientific & Technology Research, 8, 602-606.

[8] Zhang, X., & Yue, W. T. (2020). Integration of on-premises and cloud-based software: the product bundling perspective. Forthcoming in Journal of the Association for Information Systems.

[9] https://www.gartner.com/en/documents/4006317

[9] Cestari, R. H., Ducos, S., & Exposito, E. (2020, September). IPaaS in agriculture 4.0: an industrial case. In 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 48-53). IEEE.

[10] Neifer, T., Lawo, D., Bossauer, P., & Gadatsch, A. (2021, July). Decoding IPaaS: Investigation of User Requirements for Integration Platforms as a Service. In ICE-B (pp. 47-55).

[11] Frantz, R. Z., Corchuelo, R., Basto-Fernandes, V., Rosa-Sequeira, F., Roos-Frantz, F., & L. Arjona, J. (2021). A cloud-based integration platform for enterprise application integration: A Model-Driven Engineering approach. Software: Practice and Experience, 51(4), 824-847.

[12] Hyrynsalmi, S. M., Koskinen, K. M., Rossi, M., & Smolander, K. (2021, June). Towards the utilization of Cloud-based Integration Platforms. In 2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC) (pp. 1-8). IEEE.

[13] Hyrynsalmi, S. M. (2022, May). The State-of-the-Art of the Integration Platforms as a Service research. In 2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB) (pp. 17-22). IEEE.

[14] Sänger, N., & Abeck, S. (2022). Authentication and Authorization in Microservice-Based Applications. *INFORMATIK 2022*.

[15] Hyrynsalmi, S. M. (2022, October). Definition of the Enterprise Integration Platforms as a Service—Towards a Common Understanding. In Software Business: 13th International Conference, ICSOB 2022, Bolzano, Italy, November 8–11, 2022, Proceedings (pp. 167-181). Cham: Springer International Publishing.

[16] Hyrynsalmi, S. (2022). Pathway to the successful integration platform management.

[17] Hyrynsalmi, S., & Smolander, K. (2023). Between the Rock and the Hard Place-Conflicts in Implementing Integration Platforms.

[18] Huang, R. (2023, April). Research on technical path and practice of digital transformation of colleges based on iPaas. In Second International Conference on Digital Society and Intelligent Systems (DSInS 2022) (Vol. 12599, pp. 269-275). SPIE.

[19] Dahl, O. (2002). Enterprise application integration. School of Mathematics and Systems Engineering.

[20] Menge, F. (2007, August). Enterprise service bus. In Free and open source software conference (Vol. 2, pp. 1-6).

[21] Soomro, T. R., & Awan, A. H. (2012). Challenges and future of enterprise application integration. International Journal of Computer Applications, 42(7), 42-45.

[22] Goel, A. (2006). Enterprise integration EAI vs. SOA vs. ESB. Infosys Technologies White Paper, 87.

How do different communication protocols influence service discovery in microservice architectures

Haochen Huang
Universiteit van Amsterdam
012 WX Amsterdam, Netherlands
peter.huang@student.uva.nl

Kaiwei Chen
Universiteit van Amsterdam
012 WX Amsterdam, Netherlands
kai.chen@student.uva.nl

Zongyao Zhang
Universiteit van Amsterdam
012 WX Amsterdam, Netherlands
zongyao.zhang@student.uva.nl

Abstract

Microservice architecture(MSA) is a software architecture commonly used in industry nowadays, and service discovery, as an important part of microservice architecture, plays a key role in the microservice framework. This paper discusses the impact of HTTP/REST, gRPC, and MQTT on microservice architecture from the perspective of communication protocols, and gives suggestions for choosing communication protocols under different development requirements of microservice architecture.

1 Introduction

1.1 Micro-services

The first thing we need to understand is what microservices architecture(MSA) is. MSA is characterized by the development of a single application as a set of small, independent services, each running in its own process and communicating through lightweight mechanisms. These services are built around specific business capabilities and can be deployed independently through automated processes. They have minimal centralized management and can be developed using different programming languages and data storage technologies.Fowler and Lewis [2014] In addition to MSA, the industry commonly uses monolithic architectures, two different software development approaches. In the monolithic architecture, the entire application is developed as a single, independent unit. All components of the application are tightly coupled and share the same code base, database, and memory space. A monolithic architecture is easier to develop and deploy, but as an application grows, it can become complex and difficult to maintain. Extending a monolithic application requires extending the entire application, which can be inefficient and expensive. In a microservices architecture, however, applications are broken down into smaller, independent services that can be developed, deployed and scaled independently. Each service is responsible for a specific task and communicates with other services through APIs. The difference of those two architecture can be seen in Figure1 Microservices architectures are more flexible and scalable than monolithic architectures because each service can scale independently based on its specific needs. However, the development and deployment of a microservices architecture can be more complex because it requires the management of multiple services and their interactions. With such a need for interaction and communication, service discovery becomes particularly important in it.Al-Debagy and Martinek [2018]

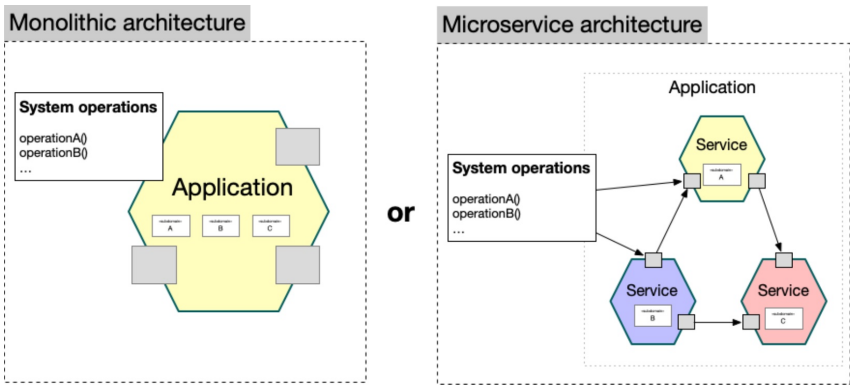


Figure 1: Two types of architecture

1.2 Service discovery

In MSA, finding services is a difficult task. To communicate with other services in a microservice architecture, a service must find out where they are located. In terms of development, testing, deployment, and horizontal scaling, monolithic deployments are straightforward. Application development, however, becomes more challenging as they grow larger and more complicated, making agile development and delivery impossible. However, MSA, a novel paradigm, holds considerable promise in terms of service complexity, adaptability, modularity, and scalability. The majority of modern microservices-based applications run in virtualized environments, where virtual service instances are transportable, decoupled, and can have dynamically changing locations given to them. Since each service in these contexts must find active instances of other services, service discovery is now a crucial component of MSA. Finding the location of the proper instance that offers the necessary service is the duty of the service discovery mechanism. Fixing the address of the service instance in the application largely resolves the service discovery issue in monolithic architectures. Fixing addresses is still not possible because services in MSAs are transient. As a result, MSA now includes agile service discovery mechanisms as a key component. Client-side and server-side service discovery are the two basic categories of service discovery patterns. The two pipelines can be seen in Figure2. First, the client service is in charge of figuring out where on the network its service dependencies are located. It accomplishes this by leveraging a service registry, a repository of accessible service instances. The client service makes a request through a router or load balancer, which checks the service registry and directs each request to an available service instance, in order to perform server-side service discovery. Both models make use of a service registry, which has to be updated often in order to reflect the service instances that are currently accessible appropriately. A service instance registration mechanism is typically used to carry out this update, and representatives are registered and deregistered as they go online and offline, respectively.

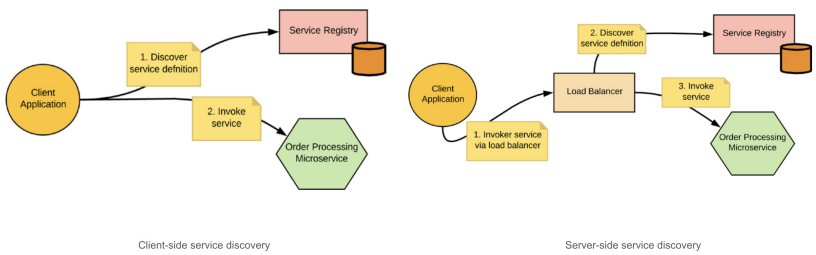


Figure 2: Two main types of service discovery patterns

1.3 Understanding Communication Protocols

The first question we need to understand is what communication protocol. A communication protocol is a system of rules that allows two or more entities of a communication system to transfer information through any change in physical quantities. The protocol defines the rules, syntax, semantics, and synchronization of the communication as well as the possible methods of error recovery. Protocols can be implemented by hardware, software, or a combination.Hilpisch et al. [2009]

1.3.1 HTTP/REST

The World Wide Web's primary data transfer protocol, Hypertext Transfer Protocol (HTTP), and the architectural style Representational State Transfer (REST), which specifies a set of limitations for developing Web services, have made significant inroads in web technologies. In-depth explanations of HTTP and REST's key characteristics, functional principles, advantages, and potential applications are provided in this literature review. The formatting guidelines for the NeurIPS 2020 conference have been followed in preparing this article.

As a global standard for data transfer over the internet, HTTP was created. It allows users to communicate and share information through web pages and functions as a request-response protocol in a client-server computing model. Fundamentally, HTTP uses a collection of standardized techniques to launch requests and responses, facilitating a smooth interchange of information on a worldwide scale. It has become the de facto protocol for web-based data exchange due to its widespread use and easy interaction with other web technologies.

The structure of HTTP as shown in Figure 3. Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called requests and the messages sent by the server as an answer are called responses. On the other hand, REST introduced an architectural style that lays the groundwork

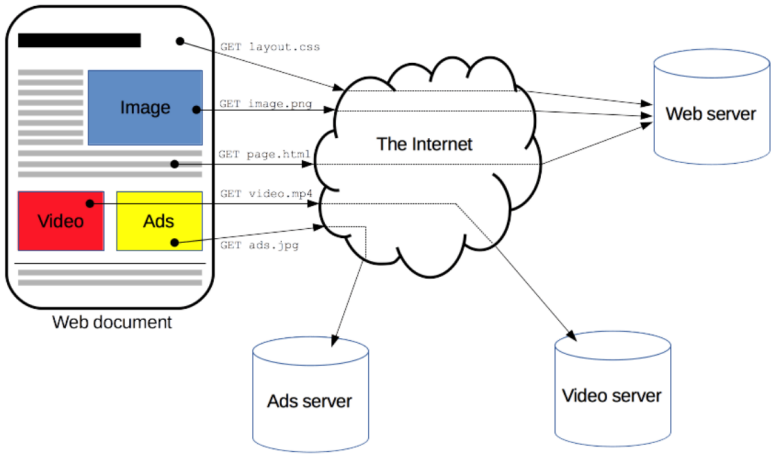


Figure 3: The Structure of HTTP.

for developing logical and simple-to-understand APIs, revolutionizing how developers design web services. REST's reliance on statelessness, which requires that each HTTP request from a client to a server be self-sufficient and convey all the necessary information to grasp and process the request independently, is one of its guiding principles. This paradigm promotes statelessness in API architecture, which supports an environment conducive to scalability and modularity principles.

RESTful APIs' inherent statelessness, which enables high scalability and makes them suitable for massive, distributed systems, is one of their most appealing features. Due to its statelessness, REST can handle numerous requests simultaneously, making it ideally suited for today's web applications, which are naturally dispersed and call for scalability. Statelessness also encourages more manageable sessions, improves server speed, and improves the user experience overall by offering a quicker and more dependable service.

RESTful is a set of resources that include a MIME type (such as JSON or XML), a primary URI for accessing the service, and a list of pre-defined operations like HTTP GET, HTTP POST, HTTP PUT, HTTP PATCH, or HTTP DELETE. Unlike web services, RESTful services are not subject to any specified standards. REST is an architectural aesthetic rather than a set of rules. The interaction between a RESTful API and clients or consumers to carry out read, add, and modify activities against a database is shown in Figure 4.

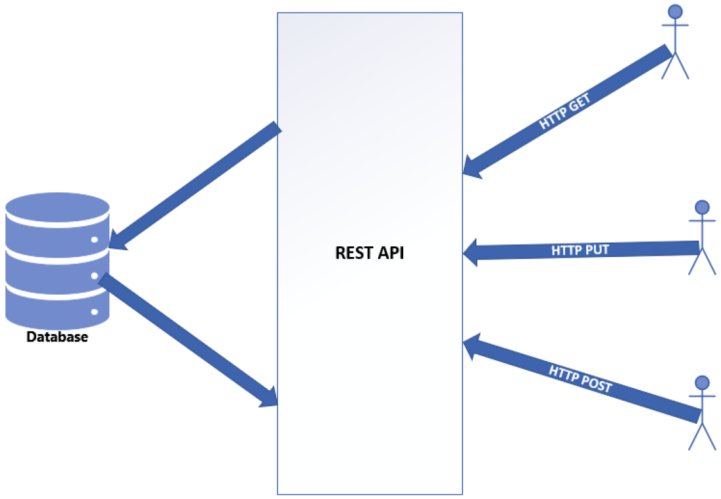


Figure 4: The Structure of REST.

1.3.2 RPC

The Remote Procedure Call (RPC) Protocol [Birrell and Nelson, 1984] is a widely known message exchange mechanism for distributed systems. The low overhead, ease of use, and transparency of the RPC protocol make it the best choice for inter-device communication in complex distributed systems containing a large number of devices [Lee, 1998]. the RPC allows the local machine to call the services of different remote servers as if they were owned by the local machine itself. Figure 5 illustrates the framework of the RPC protocol. Stubs are the most important element in the RPC framework because they enable communication between the client and the server. During an RPC call, the client needs to call a stub to pass the function to be called, the type of parameters, the names of the parameters, etc. into the stub and to serialize the message to be transmitted to the server over the network. The server-side stub is responsible for deserializing these parameters and calling the corresponding server-side function to return the result to the client. It can be said that the stub performs most of the functions of the RPC protocol. The existence of the stub allows the client to focus on function invocation without having to deal with network communication with the server. The definition of a service in an RPC server is determined by the request and response messages and this includes both Unary RPC services and Streaming services [Lu, 2020]. The mechanism of a unary RPC service is similar to a function call containing a request-response message pair, while a streaming service allows multiple messages to be sent and received between the client and the server. In addition, the message exchange mechanism of the RPC protocol is a typical synchronous type because when the client sends a new request to the server the response process will be blocked until a message is received back from the server. For asynchronous message exchange mechanisms such as the MQTT protocol see Section 1.5. For an RPC framework, the interface definition language (IDL) file is one of the essential elements for using the framework. The IDL file is used to define the interface between the client and the server and is written in a language independent of the programming language [Ford et al., 1995]. The role of the IDL file is to generate stubs for the client and the server via a stub generator. The IDL file is therefore the author of the client-server communication specification. Without the IDL file, the client would not be able to call the desired service correctly or even find the address of the server. It can be argued that the key to the operation of the entire RPC process is the writing of the IDL file, which in turn connects the IDL to

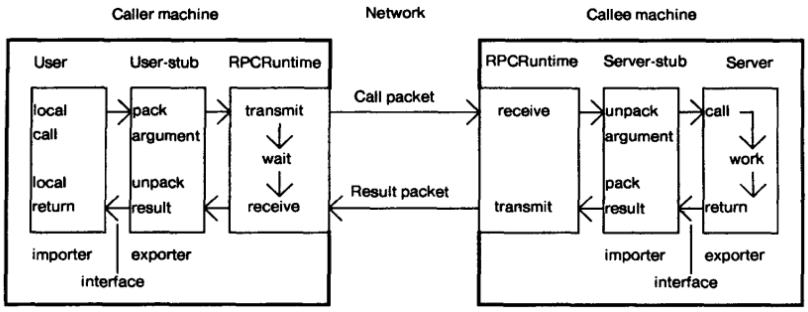


Figure 5: The components within the RPC.

the programmer.The programmer needs to define in the IDL file what services can be called by the client and what parameters need to be passed.

Although the RPC protocol was invented a long time ago for the exchange of information between client and server it does not mean that it is obsolete. In today's popular MSA, RPC is still widely used to communicate between different services [Huang et al., 2020]. gRPC is a modern RPC framework (client/server model) based on the HTTP/2 protocol developed by Google. gRPC's interface definitions are written using the Protocol Buffer library and stored in .proto files (a kind of IDL file). gRPC has several advantages over traditional RPC frameworks. Firstly, thanks to the multiplexing features of HTTP/2 gRP is able to deliver more information with less network connection overhead than traditional RPC frameworks using HTTP/1.x. Secondly, gRPC uses a Protobuffer as the storage format for serialized data, which reduces the size of the data significantly compared to textual formats such as JSON [Popić et al., 2016]. In addition, binary data is preferred to text for network transmission. Finally, gRPC supports cross-language development such as C++, Java, and Python, which allows developers to write clients and servers in different programming languages. With all these advantages, the gRPC framework offers outstanding performance and flexibility.

1.3.3 MQTT

Message Queue Telemetry Transport (MQTT)Soni and Makwana [2017]is a lightweight and bandwidth-efficient protocol used for communication between devices in the Internet of Things (IoT). It was invented by Andy Stanford-Clark of IBM and Arlen Nipper of Cirrus Link Solution. It works by using a publish/subscribe model, where devices can publish messages to a broker, which then distributes those messages to other devices that have subscribed to receive them. This allows for efficient and scalable communication between devices with limited resources.The three components can be seen in Figure6.

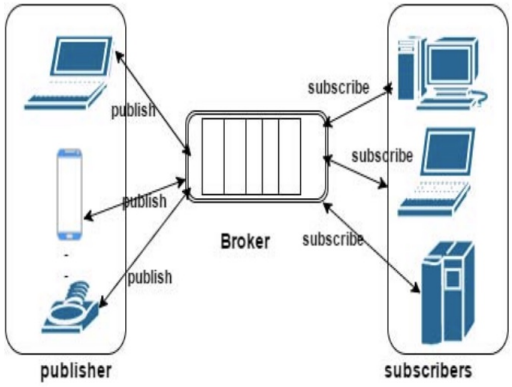


Figure 6: The architecture of MQTT.

From the figure we can see that the publish/subscribe pattern is a messaging pattern used in MQTT and is central to its function. This pattern is divided into three primary components: the publisher, the subscriber, and the broker.

The first part of it is the publisher. It is the component that produces the message. The publisher does not send the message directly to a specific subscriber. Instead, it categorizes the message under a specific topic and sends it to a broker. For the second part of it the architecture, it is the broker that becomes the intermediary that receives all messages from the publishers. It sorts these messages based on their topics. The subscriber receives messages based on specific topics it has subscribed to. The subscriber does not know the identity of the publisher. The broker sends all messages that fall under the subscriber's topics of interest.

1.4 Scope

For the scope of our research, we expect to take a deep dive into communication protocols in microservices to understand how they affect the efficiency of service discovery. Our point study focuses on four specific communication protocols: HTTP/REST, gRPC and MQTT. hopefully, at the end of our research, it will provide some useful insights into choosing communication protocols in a microservices architecture.

2 Influence of Communication Protocols on Service Discovery Efficiency

2.1 HTTP/REST

Service discovery has become a critical aspect in the world of HTTP/REST services due to the extensive increase in the variety and number of services available. The efficiency of service discovery plays a significant role in web-based systems' overall performance and usability. To understand the influence of HTTP/REST on service discovery efficiency, we draw upon insights gleaned from three research papers and a book as outlined.

Elshater et al. [2015] present an innovative technique for discovering web services called "goDiscovery." This approach exploits the fundamentals of HTTP/REST and facilitates efficient discovery by enabling clients to find and integrate with the necessary services quickly. The "goDiscovery" approach takes advantage of REST's statelessness, providing quick responses and reducing the overall latency in the service discovery process. The inherent properties of HTTP/REST protocols are leveraged to enhance the efficiency of service discovery. What is more, Aziez et al. [2019] present a comparative study of service discovery approaches in the Internet of Things (IoT) context. The authors highlight that HTTP/REST-based service discovery approaches perform exceptionally well due to their lightweight nature and statelessness, which makes them apt for IoT applications. The ability of RESTful APIs to handle requests concurrently also allows for more efficient discovery of services in an IoT context, which often involves thousands of devices. Moreover, Verborgh et al. [2011] introduced a novel "Hyperlinked RESTdesc" approach for runtime service discovery. It leverages hypermedia controls provided by RESTful HTTP services to aid in efficient service discovery. The paper underscores that the hypermedia-driven nature of HTTP/REST protocols allows for improved navigability and discoverability of services at runtime. And Hutchison [2015] address the challenges of engineering web applications in the significant data era and discusses the role of HTTP/REST in enhancing service discovery. It emphasizes that the scalable nature of HTTP/REST makes it conducive for managing and discovering services in big data applications, as it can easily handle an enormous volume of data and services.

A compelling claim is made by combining the knowledge gained from several studies: HTTP/REST protocols fundamentally improve the effectiveness of service discovery. The processes that speed up the identification and integration of web services are part of their fundamental traits, far outperforming the effectiveness of traditional approaches. Their architecture, which is naturally lightweight and capable of handling many requests, enables distributed systems to communicate more quickly. It is instrumental in settings with high data volumes or network traffic.

Because of their love for hypermedia, HTTP/REST protocols encourage increased runtime navigability. Utilizing hypermedia controls enables dynamic service traversal, facilitating the

quick finding of valuable resources. This characteristic guarantees seamless service interaction, enables effective integration and promotes a vast and interconnected system. In large-scale systems, the impact of HTTP/REST increases its power in scalability and performance. These protocols provide improved scalability since they are stateless, a quality crucial in the Big Data era. Under the supervision of HTTP/REST protocols, large-scale applications and services can communicate and handle enormous volumes of data and concurrent services while retaining efficacy.

Moreover, HTTP/REST protocols are well positioned in the Internet of Things (IoT) environment due to their ability to handle concurrent queries and lightweight architecture. These characteristics provide simplified data exchange in a system with an extensive network of devices that demands effective communication, improving service discovery performance.

To sum up, HTTP/REST protocols have a profound and multifaceted impact on the effectiveness of service discovery, influencing operational capabilities across a wide range of applications such as web-based systems, IoT, and Big Data solutions. Their contributions improve these domains' overall agility, resilience, and effectiveness. The importance of HTTP/REST protocols and their influence on service discovery is expected to grow. They play a crucial role in this environment because of their innate capacities to promote effective service discovery, support robust web systems, and power durable IoT applications.

2.2 RPC

As described in section 1.4, the RPC protocol requires the client to know many details such as function names, parameters, and even addresses before calling the server. Therefore, the client and the server need to be pre-bound in order to exchange information. However, the client is most interested in using the desired service and not in all the details of the service. It can be argued that the use of RPC reduces the abstraction of the system architecture and thus requires the client and even the programmer to master too many details.

The research by Jacob and Mudge [1996] points out the incompetence of RPC protocols in a nomadic computing environment. Nomadic computing requires mobile devices to be able to discover and use new services as they move to new environments [Kleinrock, 1995]. The authors argue that there are two influential factors in order to achieve nomadic computing. Firstly, the authors suggest that the client should be able to use something like a directory to find the service-based properties of a service when entering a new environment. Secondly, the authors consider that the client must have the ability to dynamically acquire the service interface, i.e. instead of using a stub generated from an IDL file to predefine the interface between the client and the server, the interface description must be obtained directly from the server. For a solution, the authors first discuss the option of modifying the RPC protocol to use a protocol interpreter to interpret the new service interface in order to eliminate the client's reliance on the IDL file, however, they conclude that this would break the structure of the RPC protocol itself. The authors then propose a standard for supporting nomadic computing called service discovery. Service discovery, as defined by the authors, requires allowing a client to retrieve a list of services from a service catalog and connect to a server and then obtain an interface directly from the server and interact with it. From the above mechanism, it appears that what Jacob and Mudge propose is the prototype of a modern service discovery and registration mechanism.

A number of alternative solutions have been proposed to address the need to explicitly define function calls and references to the server in the use of the RPC protocol. The Cygnus model is a model for abstracting services [Chang et al., 1991]. The core idea of the Cygnus model is to decouple the service from the server and even the service interface so that the client gets a high-level view of the corresponding service. The Cygnus model consists of four elements, the client, the service types, the service entities, and the server. In the Cygnus model, the client accesses the desired service through a set of attributes, i.e., the service type. For example, a set of attributes may contain the name of the service and the version of the service. The service entity represents the service interface in the Cygnus model and the model binds the service type to the service entity rather than to the server because the server may expose a number of interfaces for different services. The key point in the Cygnus distributed system is the use of a distributed database because the database maintains the relationship between the service types, service entities,

and servers. Querying the database is a necessary step in the Cygnus system to obtain the true service implementer. Research on the Cygnus model has also shown that the Cygnus system has a lower overhead than Sun RPC (an RPC framework). The key contribution of this research is the proposed method of abstracting the service, thus reducing the amount of detail that the client needs to know about the server. However, the problem is not fully solved as the Cygnus model still requires the client to learn the service interface in advance (it cannot be implicitly bound to the server).

Similarly, in their research, ord Neuman et al. [1993] propose the use of the Prospero directory service to solve the problems in pervasive computing. The concept of pervasive computing is similar to that of nomadic computing. The two most important characteristics of pervasive computing are mobility and scale. Mobility means that when a user's location changes the user may need to choose another server to get the corresponding service. Scale means that the number of services may overload the user as the user's location changes because the user needs to maintain a large amount of information about the service. Therefore, the two biggest problems of pervasive computing are the server selection problem and the user location problem. To solve the server selection problem, Prospero allows users to create virtual systems for mapping names to servers. The biggest advantage of Prospero is that mappings can be dynamically determined through virtual system aliases, union links, and filters. The advantages are very similar to the Cygnus model of invoking services by service type as they do not require the provision of service specifics but only some properties of the service. The issue of user location is mainly concerned with the storage of user login information in a distributed system and is not relevant to service discovery and is therefore not discussed here.

One way used to resolve implicit client-server binding is by inserting RPC agents between the client and the server [Huang and Ravishankar, 1994]. Using RPC agents will transfer the responsibility for learning the interface from the client to the agent. In this way, the problem of different RPC protocols being used between the client and the server can also be solved since they would not be able to communicate directly. The authors propose two agent models in their research, the one-agent model and the two-agent model. The one-agent model converts communication between different RPC protocols by deploying an agent between the client and the server. The two-agent model deploys one agent on the client and one on the server and uses the proxies to connect and communicate. In the study, the authors also discuss the advantages and disadvantages of the two agent models. The authors of the study point out that the two-agent model is better because it reduces the complexity of agent construction. For the one-agent model, the agent needs to learn the client and server RPC protocols, whereas for the two-agent model, the agent only needs to learn the protocols of the local and agent links. The one-agent model makes it necessary for the agent to exist on either the client or the server, thus making it necessary for the agent to learn non-local RPC protocols. In terms of performance, research has shown that although the two-agent mode causes some performance degradation, it is acceptable (2 ms to 6 ms). However, there are some drawbacks to this approach. Firstly, although the responsibility for learning the interface is transferred to the agent, the construct becomes more complex resulting in additional overhead. The agent needs to be updated to learn the new interface and the client needs to invoke the service according to the interface specified by the agent. The second problem is that the client still needs to know the details of the information it wants to send to the agent, thus leaving the abstraction problem behind.

Modern RPC frameworks such as gRPC still do not seem to address the issue of dynamically learning interfaces. Figure 7 illustrates the structure of gRPC. From the above picture, we can see that gRPC allows different programming languages to be used to write the client and server but the client still needs to use the stubs to communicate with the server. gRPC indeed requires the relevant information to be written into the IDL file for use. This means that the client and server must adhere to the same interface for information exchange rather than allowing the client to learn the interface description from the server as proposed by the service discovery standard. Furthermore, gRPC does not have an embedded service discovery and registration mechanism. gRPC's built-in service discovery only supports DNS resolution, i.e., finding the corresponding IP address by name. However, gRPC has the ability to implicitly bind clients and servers by using third-party registry components such as ZooKeeper and etcd. Using the registry clients can dynamically look up the IP address of the required service without the need to specify service-related properties prior to the call.

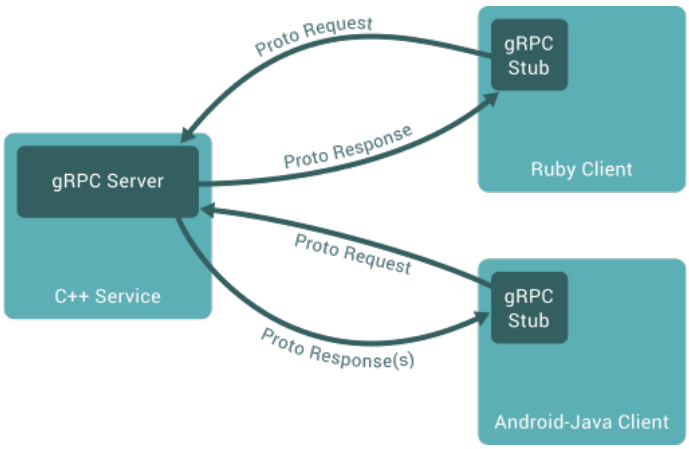


Figure 7: The structure of gRPC [Google, 2023].

2.3 MQTT

For the service discovery of MQTT. It is mainly discussed under the IoT background. While MQTT can be used to exchange information about available services, it is not specifically designed for service discovery and there surely have some drawbacks when coming to the service discovery part. MATIC et al. [2021] Geonwoo Kim et al. [2019]mentioned that the autoconfiguration feature of it could let it act bad in multicasting and resource directories for managing IoT resources while using MQTT can help optimize communication between cloud microservices in the IoT cloud. While MQTT does offer different QoS (Quality of Service) levels, the guarantees it provides at each level may not be sufficient for some applications.Sadeq et al. [2019] For instance, QoS ensures that messages are delivered exactly once, but it involves a complex four-step handshake which can be overkill for simple sensor readings. Another drawback of it is the single point of failure. MQTT relies on a central broker to relay messages. If this broker fails, the entire messaging system can come to a halt. Hence, it's critical to ensure the broker is highly available and scalable. There is another threat that while MQTT does support TLS/SSL for secure message transmission, it lacks built-in support for more complex security measures like message signing or encryption, access control, and other common security requirements in a microservice architecture.Iyer et al. [2018] On the other hand, MQTT's lightweight publish/subscribe communication model could be leveraged to create a system where services can dynamically publish their presence or subscribe to other services especially when it comes to IoT scenarios and resource Constrained Environments.

3 Discussion and Conclusions

Micro-service Architecture	**Choose of Protocol**	**Reasons**	**Possible Issues**
Performance	gRPC	Protocol Buffers, HTTP 2.0, Multiple Language Support	Compatibility (Network/Language), Service Discovery
Structures an application as a collection of loosely coupled, independently deployable services	HTTP/REST	Stateless nature, interoperability	Add overhead, lack real-time communication capabilities
IoT Scenarios, Need for Publish/Subscribe Model	MQTT	Designed with IoT, Resource Constrained Environments,Need for Publish/Subscribe Model	Limited QoS Single Point of Failure Lack of Standard Security Mechanisms

When we examine the various communication protocols and their ramifications, as shown in the table above, we can see the delicate role of communication protocols in influencing service discovery within MSA. The research articles "Evaluating the Impact of Inter-Process Communication in Microservice Architectures" and "Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: an energy cost analysis" offer insightful information in this regard.

The gRPC protocol is frequently selected for high-performance MSA because of its use of Protocol Buffers and HTTP 2.0 and its wide range of language compatibility. The research "Evaluating the Impact of Inter-Process Communication in Microservice Architectures" explains how gRPC beats its rivals thanks to its effectiveness in offloading computation and lower energy cost. It also draws attention to potential problems, such as the difficulty of service discovery and interoperability with various network and linguistic infrastructures. The difficulty of service discovery increases due to gRPC's non-human readable data format, highlighting the necessity of effective service discovery procedures in such situations.

Due to its statelessness and interoperability, HTTP/REST is frequently chosen by microservices that want to be loosely connected and independently deployable. This makes service discovery and interaction more straightforward. Because each request can be handled separately because of the protocol's statelessness, service discovery in distributed systems is made more accessible. However, using HTTP/REST can be inefficient for some application scenarios because it adds overhead and does not support real-time communication.

The MQTT protocol is a top pick for IoT applications when the requirement for a Publish/Subscribe paradigm is critical. Shafabakhsh et al. [2020] explores how MQTT's design is specifically adapted to IoT and resource-constrained settings. The article also highlights the protocol's shortcomings, such as low Quality of Service (quality of service), single points of failure, and a lack of standardized security measures, which may impact service discovery effectiveness in certain circumstances.

Chamas et al. [2017] emphasized how crucial it is to pick the proper protocol based on the application's particular needs. The protocol choice regarding energy costs, computing efficiency, and other factors can impact the efficiency of service discovery.

Finally, there are numerous ways in which differing communication protocols affect service discovery in MSA. Variables, including the type of microservices, the context of the application, and the unique properties of the protocols themselves, greatly influence the effectiveness of service discovery. To successfully navigate the world of MSA, it is, therefore, essential to have a thorough understanding of these protocols and their ramifications.

References

O Al-Debagy and P Martinek. A comparative review of microservices and monolithic architectures. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 000149–000154, 2018. doi: 10.1109/CINTI.2018.8928192.

Meriem Aziez, Saber Benharzallah, and Hammadi Bennoui. A full comparison study of service discovery approaches for internet of things. *International Journal of Pervasive Computing and Communications*, 15(1):30–56, 2019.

Andrew D Birrell and Bruce Jay Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, 2(1):39–59, 1984.

Carolina Luiza Chamas, Daniel Cordeiro, and Marcelo Medeiros Eler. Comparing rest, soap, socket and grpc in computation offloading of mobile applications: An energy cost analysis. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE, 2017.

Rong N Chang, Chinya V Ravishankar, and Jacob Slonim. A service acquisition mechanism for the client/service model in cygnus. In *CASCON*, pages 323–345, 1991.

Yehia Elshater, Khalid Elgazzar, and Patrick Martin. godiscovery: Web service discovery made efficient. In *2015 IEEE International Conference on Web Services*, pages 711–716, 2015. doi: 10.1109/ICWS.2015.99.

Bryan Ford, Mike Hibler, and Jay Lepreau. Using annotated interface definitions to optimize rpc. In *Proceedings of the fifteenth ACM symposium on Operating systems principles*, page 232, 1995.

M Fowler and J Lewis. Microservices, March 2014. URL `http://martinfowler.com/articles/microservices.html`.

Google. Introduction to grpc, 2023. URL `https://grpc.io/docs/what-is-grpc/introduction/`.

R E Hilpisch, R Duchscher, M Seel, et al. Wireless communication protocol, May19 2009. URL `https://patents.google.com/patent/US7529565B2/en`. U.S. Patent No. 7,529,565.

Longda Huang, Weijin Zhuang, Mingyang Sun, and Hong Zhang. Research and application of microservice in power grid dispatching control system. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 1895–1899. IEEE, 2020.

Yen-Min Huang and Chinya V Ravishankar. Designing an agent synthesis system for cross-rpc communication. *IEEE transactions on software engineering*, 20(3):188–198, 1994.

David Hutchison. *Engineering the Web in the Big Data Era*, volume 9114. 2015. ISBN 978-3-319-19889-7.

Shweta Iyer, G. V. Bansod, Praveen Naidu V, and Shefali Garg. Implementation and evaluation of lightweight ciphers in mqtt environment. In *2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 276–281, 2018. doi: 10.1109/ICEECCOT43722.2018.9001599.

Bruce Jacob and Trevor Mudge. Support for nomadism in a global environment. In *Proc. Workshop on Object Replication and Mobile Computing*, 1996.

Geonwoo Kim, Seongju Kang, Jiwoo Park, and Kwangsue Chung. An mqtt-based context-aware autonomous system in onem2m architecture. *IEEE Internet of Things Journal*, 6(5):8519–8528, 2019. doi: 10.1109/JIOT.2019.2919971.

Leonard Kleinrock. Nomadic computing—an opportunity. *ACM SIGCOMM Computer Communication Review*, 25(1):36–40, 1995.

Jong-Kun Lee. A group management system analysis of grpc protocol for distributed network management systems. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 3, pages 2507–2512. IEEE, 1998.

Zonghao Lu. A case study about different network architectures in federated machine learning, 2020.

Milica MATIC, Marija ANTIC, Istvan PAPP, and Sandra IVANOVIC. Optimization of mqtt communication between microservices in the iot cloud. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–3, 2021. doi: 10.1109/ICCE50685.2021.9427602.

B Cli ord Neuman, Steven Seger Augart, and Shantaprasad Upasani. Using prospero to support integrated location-independent computing. In *Proceedings of the Usenix Symposium on Mobile and Location-Independent Computing*, 1993.

Srđan Popić, Dražen Pezer, Bojan Mrazovac, and Nikola Teslić. Performance evaluation of using protocol buffers in the internet of things communication. In *2016 International Conference on Smart Systems and Technologies (SST)*, pages 261–265. IEEE, 2016.

Abdulrahman Sameer Sadeq, Rosilah Hassan, Salah Sleibi Al-rawi, Ahmed Mahdi Jubair, and Azana Hafizah Mohd Aman. A qos approach for internet of things (iot) environment using mqtt protocol. In *2019 International Conference on Cybersecurity (ICoCSec)*, pages 59–63, 2019. doi: 10.1109/ICoCSec47621.2019.8971097.

Benyamin Shafabakhsh, Robert Lagerström, and Simon Hacks. Evaluating the impact of inter process communication in microservice architectures. In *QuASoQ@ APSEC*, pages 55–63, 2020.

D Soni and A Makwana. A survey on mqtt: a protocol of internet of things (iot). In *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*, volume 20, pages 173–177, April 2017.

Ruben Verborgh, Thomas Steiner, Davy Van Deursen, Rik Van de Walle, and Joaquim Gabarró Vallés. Efficient runtime service discovery and consumption with hyperlinked restdesc. In *2011 7th International Conference on Next Generation Web Services Practices*, pages 373–379. IEEE, 2011.

What are the emerging trends and future directions in multi-cloud management?

Daniele Quartinieri
Department of Computer Science
Universiteit van Amsterdam (UvA)
daniele.quartinieri@student.uva.nl

Farimah Mohebi
Department of Computer Science
Universiteit van Amsterdam (UvA)
farimah.mohebi@student.uva.nl

Daniel Garcia
Department of Computer Science
Universiteit van Amsterdam (UvA)
daniel.garcia.gaviria@student.uva.nl

June 5, 2023

Abstract

Presently, cloud computing is widely regarded as the preferred approach for addressing service execution requirements. Nevertheless, there exist multiple methods for implementing cloud computing, with each approach being optimal for specific needs and necessitating particular evaluations. The paper commences with a comprehensive introduction to cloud service and deployment models, followed by an analysis of the advantages and disadvantages of cloud computing, ultimately culminating in a discussion of Multi-cloud. The present study centers on the current status, challenges, and prospective advancements of Multi-cloud, with a particular emphasis on the management complexities and developments associated with it.

1 Introduction

Cloud computing is defined as "a way to architect and remotely manage computing resources" [28]. Through the latest twenty years cloud computing has gained more and more popularity, this can be noticed by the **Figure 1** showing AWS,

one of the most famous CSP's (cloud service provider) revenues from Q1'14 - Q3'22 [23].

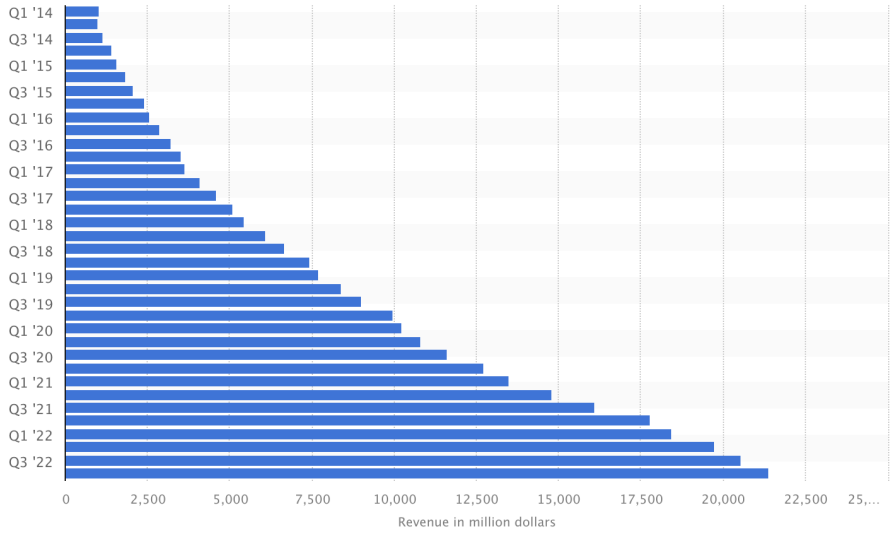


Figure 1: AWS Revenue from Q1'14 - Q3'22

Cloud computing can be provided through various service and deployment approaches.

Cloud computing service models can be categorized into three main types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [9]. According to [8] **Software as a Service** (SaaS)"the consumer can operate a software on the provider's cloud system, typically over the Internet". Based on the cited source, in the context of **Platform as a Service** (PaaS), "the provider allows the consumer to use the cloud network as a platform for their own developed or acquired programs." The **Infrastructure as a Service** (IaaS) model involves the provision of virtual storage and machines to the consumer[8].

Based on [13], Cloud deployment models/types can be categorized as follows: A **private cloud** is a type of cloud infrastructure that is allocated for the use of a single organization, which may consist of multiple consumers such as business units. The term **community cloud** refers to a cloud computing infrastructure that is allocated for the sole use of a particular community of consumers, consisting of organizations that share common concerns. The term **public cloud** refers to a cloud computing model where the cloud infrastructure is made available for unrestricted use by the general public. The concept of **hybrid cloud** pertains to a cloud infrastructure that comprises multiple, separate cloud infrastructures such as private, community, or public clouds. These individual entities are integrated through standardized or proprietary technology, which facilitates data transfer and applications across the different clouds. An example

of this is cloud bursting, which allows for load balancing between clouds.

To determine the appropriate cloud deployment model, it is necessary to assess the cloud based on the specific requirements of the final product. According to the summary provided by [17], the advantages of cloud computing can be delineated as follows:

The ability to "**Satisfy business requirements on demand**, by resizing the resource occupied by application to fulfill the changing the customer requirements", The ability to achieve "**Lower cost and energy saving**, by making use of low cost PC, customized low power consuming hardware and server virtualization, both CAPEX and OPEX are decreased" and The ability to "**Improve efficiency of resource management**, through dynamic resource scheduling"

According to [17], the drawbacks of cloud computing can be summarized as follows: The **issue of privacy and security** is a significant concern for customers who prioritize the safeguarding of their personal information and data security over conventional hosting services. The term "**continuity of service**" pertains to various factors that may potentially impede the seamless operation of cloud computing, including but not limited to issues with internet connectivity, power outages, service interruptions, and system glitches. Presented below are some common instances of such issues: In November of 2007, RackSpace, a competitor of Amazon, experienced a service disruption lasting three hours due to a power outage at its data center. Similarly, in June of 2008, Google App Engine service experienced a six-hour interruption due to storage system bugs. Finally, in March of 2009, Microsoft Azure encountered a 22-hour service outage caused by an operating system update. The present public cloud provider, which operates on virtualization, stipulates a service reliability of 99.9% in its Service Level Agreement (SLA). In addition, the **migration of services** represents a significant challenge within the context of cloud computing. There is a lack of consensus among regulatory bodies regarding the standardization of the external interface of cloud computing.

In order to address certain apprehensions associated with cloud computing, such as those pertaining to privacy, security, and continuity of service, various hosting infrastructure strategies, such as **Hybrid-cloud** and **Multi-cloud**, may be implemented.

The concepts of multi-cloud and hybrid cloud both involve integrating multiple cloud services. However, the distinguishing factor between the two lies in the deployment type of the cloud. Specifically, hybrid-cloud pertains to the utilization of two or more distinct types of cloud (i.e., public or private), while multi-cloud pertains to the utilization of various clouds from different vendors, all of the same type (i.e., public or private) [26][14]. Adopting a multi-cloud strategy may entail the utilization of either two public cloud infrastructures or two private cloud infrastructures. The implementation of a hybrid cloud strategy entails the utilization of both a public cloud infrastructure and a private cloud infrastructure[26].

2 Literature / background

2.1 Multi-cloud computing challenges

In reference to [7], some of the multi-cloud's challenging characteristics are the concept of **workload portability**, which refers to developing an application only once and deploying it on multiple cloud computing platforms. And the concept of **workflow portability** pertains to sustaining a uniform workflow across multiple clouds. The concept of **data portability** that refers to the ability to transfer data from one cloud platform to another. Finally, **traffic portability** that pertains to the capability of directing traffic among clients that are located in different geographic locations.

Some of these challenges are introduced by the practice of Cloud-Service-Providers' (CSP) to vendor lock. With vendor lock, we define "a situation where the cost of switching to a different vendor is so high that the customer is essentially stuck with the original vendor."[27] "Vendor lock-in problem in cloud computing is characterized by expensive and time-consuming migration of application and data to alternative providers. Cloud software vendors lock in customers in several ways: (1) by designing a system incompatible with software developed by other vendors; (2) by using proprietary standards or closed architectures that lack interoperability with other applications; (3) by licensing the software under exclusive conditions"[15]

Vendor locking can be mitigated on various levels by the adoption of non-proprietary technologies that are supported by multiple vendors/CSPs. To make some examples, on the containers and cluster level, an option could be Docker for containers and Kubernetes for clusters, while on an IaaS level, an option could be an open-source IaaS platform like OpenStack which can be public or private [19] and can be hosted by multiple CSP's.

Access management and authorizations are another Multi-cloud computing challenge. This is due to the need for operators to manage authorizations and access multiple cloud instances. A significant challenge in a multi-cloud environment is the increased responsibility placed on the organization to ensure the security of data stored across multiple cloud providers and the secure exchange of information between them. Another challenge is to support, as it is imperative that the end user is not required to be concerned with the provider responsible for delivering a service. As such, it becomes necessary to incorporate a client-support or self-service feature[2].

2.2 Multi-cloud computing benefits

The adoption of multi-cloud computing has gained significant attention among enterprises because of its diverse advantages, as outlined by [16]; these advantages include: Managing service peaks, balancing cost efficiency and service excellence, responding to modifications in provider offerings, implementation of certain limitations such as new regulations or geographical locations, ensuring the availability of resources and services, creating backups for unexpected outages

or planned maintenance, serving as an intermediary, and improving one's own cloud services and resources through agreements with third-party providers.

2.3 The current state of multi-cloud management

According to [10], due to the increasing complexity of multi-cloud architectures, it is likely that more organizations will use multi-cloud management tools in the coming years.

Based on [11], the growth of the multi-cloud management market is anticipated as businesses increasingly investigate and adopt various cloud service providers. Enterprises are expected to seek comprehensive solutions for managing their multi-cloud. In addition to the existing methodology, there will be a heightened need for platforms that facilitate the management of multiple clouds. In numerous aspects, the year 2023 has the potential to serve as the foundational year for managing multi-cloud environments, both in terms of strategic planning and operational implementation.

2.4 Multi-cloud management current solutions and trends

The usage of management tools is critical in multi-cloud environments due to the additional layer of complexity introduced to the cloud architecture, despite the numerous advantages that multi-cloud offers. The subject matter pertains to the interplay between platform heterogeneity, management control, and automated application deployment[21].

Using multi-cloud management can enable organizations to leverage the advantages of different cloud services while avoiding the challenges of managing numerous platforms. Managing multiple clouds also facilitates transparency, as the associated tools typically allow overseeing workloads and other metrics. Moreover, Multi-cloud management tools offer advantages such as aiding IT departments in implementing security policies and providing proactive assistance in identifying potential security vulnerabilities. These tools can also facilitate the management of expenses [25]. Furthermore, they allow developers to expeditiously and reliably generate applications. Multi-cloud management solutions commonly offer automated provisioning functionality and workflow management [18].

Current solutions and trends of multi-cloud management include utilizing multi-cloud tools and multi-cloud platforms.

2.4.1 DevOps tools

Greater cloud coverage can increase the potential for misconfigurations, resulting in security vulnerabilities, unanticipated system behavior, suboptimal resource allocation, and excessive monthly cloud expenditures [5].

Infrastructure-as-code (IaC) solutions, namely *Chef*, *Puppet*, *Ansible*, and *Terraform*, belong to the category of DevOps tools that generate policy-based templates to ensure uniform configuration and provisioning of cloud-based server

environments. These tools aid in mitigating the likelihood of misconfigurations and eliminate the need for conjecture when administering servers on various platforms [5]. For example, Terraform is a cloud-agnostic tool that facilitates the management of multiple providers through a single configuration. The system is capable of managing interdependent resources across multiple cloud platforms. The simplification of infrastructure management and orchestration helps operators in constructing extensive multi-Cloud infrastructures[18].

Continuous integration and continuous delivery CI/CD tools, including *Jenkins*, *GitLab*, and *Spinnaker*, facilitate the automation of the application building and deployment process across various cloud environments. These measures decrease the administrative burden. In addition, they facilitate the early detection and resolution of bugs by developers during the delivery phase, a feature that proves especially advantageous when implementing code modifications across multiple environments [5].

2.4.2 Package Management tools

Software supply chain management solutions, such as Cloudsmith, JFrog Artifactory, and GitHub Packages, offer a secure and centrally managed repository for storing and organizing the diverse artifacts of an application[5].

Application dependency management can be simplified by utilizing tools that facilitate the management of containers, scripts, and libraries. These tools can seamlessly integrate into the continuous integration/continuous deployment (CI/CD) pipeline. The aforementioned approach is especially advantageous in intricate multi-cloud scenarios, as it affords entities a prompt and effective mechanism for disseminating their software resources to diverse sites within their cloud infrastructure[5].

2.4.3 Cloud Cost Management tools

Achieving optimal cloud cost is a challenging task, even in the case of basic deployments. Using cost-optimization tools that possess multi-cloud capabilities can aid users in maximizing the value of their on-demand infrastructure by identifying the most economically efficient provider for each of their workloads. Cloud cost management platforms that offer support for multiple clouds comprise *Apptio Cloudability*, *CloudZero*, and *Flexera One*. According to [6], Cost management tools are among the most commonly utilized types of multi-cloud tools.

2.4.4 Cybersecurity tools

The security of data is a significant apprehension among end-users in multi-cloud settings. Safeguarding said environments from potential attacks and intrusions is a significant area of focus within both academic research and industrial applications[4]. In multi-cloud environments, deploying firewalls and other standard rule-based security protection solutions is insufficient to guarantee the safety of user data.

Clients have the option to utilize the internal security measures offered by individual cloud providers. However, these security measures are typically tailored to their respective platforms, rendering them inadequate for ensuring security across multiple cloud environments. In contrast, non-proprietary tools typically lack vendor specificity and can facilitate clients in centrally managing security. For example, The aforementioned solutions encompass general-purpose capabilities that cater to multi-cloud environments, such as *Lacework*, *F5*, *Cloudflare*, and *CrowdStrike*. Additionally, composite tools, such as **security information and event management (SIEM)** systems, are utilized to consolidate and scrutinize data obtained from diverse sources. And specialized tools, such as **Cloud Security Posture Management (CSPM)** and **Cloud Infrastructure Entitlement Management (CIEM)**, are utilized to monitor and analyze configurations and permissions in cloud environments.

The authors of [20] presented a solution to address security concerns in multi-cloud environments. They established the feasibility of utilizing supervised machine learning techniques to detect anomalies and categorize attacks. They utilized a widely used dataset that is accessible to the public to construct and evaluate machine learning models for the identification and classification of various types of attacks. The researchers have employed two distinct supervised machine learning methodologies, including linear regression (LR) and random forest (RF). The findings indicate that despite achieving flawless detection, the precision of categorization may still be compromised by the existence of resemblances among various attacks. The findings indicate a detection accuracy of over 99% and a categorization accuracy of 93.6%, albeit with certain attacks being uncategorized. Moreover, the authors contend that this classification can be extended to multi-cloud environments by utilizing identical machine learning methodologies.

2.4.5 Containers

Containers have recently gained significant popularity as a virtualization alternative to conventional virtual machines. Containers utilize the kernel of the host operating system (OS) to access the necessary underlying resources instead of employing a hypervisor. This attribute facilitates the duplication of containers across disparate servers featuring distinct configurations, subject to the condition that the operating system of each server employs an identical or compatible Linux kernel. Consequently, their high level of portability makes them well-suited for deployment across various cloud environments [5].

Docker is a prominent platform that is based on Linux, and it is used for the development, transportation, and execution of applications through virtualization that is container-based. The management of a number of containers within a Docker cluster can present challenges, and as a result prompting the emergence of container-centric orchestrators such as *Docker Swarm*, *Google Kubernetes*, and *Apache Mesos*. The automation of provisioning and management of intricate containerized deployments across different hosts and locations is achieved by using container-level orchestration[24].

2.4.6 Artificial Intelligence & Machine Learning

According to [12], globally, individuals utilizing cloud services are exploring advanced cloud management tools that incorporate artificial intelligence (AI) technology to facilitate the automation of cloud performance optimization and identification of anomalies. To achieve efficacy across multiple cloud environments, artificial intelligence tools necessitate a shared representation of cloud services and the provision of machine learning optimization that caters to diverse objectives. Additionally, machine learning (ML) has garnered considerable interest in cloud and multi-cloud computing, primarily due to the cost-effective ML services offered by public cloud providers. Machine learning techniques facilitate the analysis of patterns in parameters of cloud entities, thereby enabling the optimization, classification, and prediction of cloud workloads. The coherent implementation of machine learning technology across different cloud platforms necessitates a unified depiction of said entities. Obstacles to achieving optimal results include challenges such as selecting appropriate reference sets, defining distance metrics, and considering nonfunctional properties. Resolving these obstacles is imperative for completely utilizing machine learning in multi-cloud settings and augmenting cloud administration competencies.

Some examples of using AI and ML in multi-cloud management are [20] which they proposed machine learning for Anomaly Detection and Categorization in Multi-Cloud Environments, and [1] that in multi-cloud environments, an AI-driven collaborative Intrusion Detection System (IDS) is employed as the methodology; by using machine learning techniques, historical feedback data is effectively leveraged to facilitate proactive decision-making. Additionally, Utilizing a Denoising Autoencoder (DA) is a fundamental component in constructing a deep neural network, enabling the system to render determinations regarding dubious intrusions despite receiving only partial feedback. The evaluation of the model using real-life datasets indicates a high level of detection accuracy, reaching up to 95%. Artificial intelligence can also be used for resolving IoT multi-cloud scheduling because of the complexity of multi-cloud scheduling [3].

3 Discussion

3.1 Future trends in multi-cloud managements

Subsequent investigations pertaining to multi-clouds ought to prioritize the establishment and maintenance of standardized protocols. In addition to the existing literature on multi-cloud computing, further research endeavors should prioritize the integration of multi-cloud with other cutting-edge technologies such as machine learning and big data. This is because these technologies have the potential to effectively address the current challenges associated with multi-cloud computing.[22]

Based on our research for this literature study, we decided to categorize future trends in multi-cloud management into three different categories.

3.1.1 Artificial Intelligence & Machine Learning

Artificial intelligence (AI) and machine learning (ML) are predicted to be used for the purpose of automating a wider range of tasks in multi-cloud management, including workload balancing and disaster recovery. Additionally, these technologies are expected to offer more profound insights into cloud usage, performance and enhance decision-making processes in multi-cloud environments.

3.1.2 Security

With a complex multi-cloud environment, there is a natural worry about security because the number of threats is growing. The modern, digital-native world is becoming more vulnerable to cyberattacks, and a multi-cloud ecosystem needs specific types of security to protect its data and protect the privacy of its clients, vendors, and employees. In a multi-cloud environment, companies will look for ways to break down the walls between their technical and security teams[11].

Using AI and ML in multi-cloud management tools is something new, and based on what we have found, it is mostly used for security purposes, as it is mentioned before in the Multi-cloud management current solutions and trends section. It is predictable that AI and ML will be used again for enhancing security in multi-cloud management.

3.1.3 Cost management

Companies may optimize multi-cloud costs in 2023. Vendor management and multi-cloud ecosystem use will be important here. The technical teams won't be the only ones responsible for creating a culture of justifying and optimizing multi-cloud spending. Leaders of functional groups will regularly communicate the impact of underutilized cloud instances and resources. Organizations will also invest in multi-cloud management platforms and find ways to reduce their unused cloud consumption[11].

The influence of artificial intelligence in various fields is undeniable. Therefore, using artificial intelligence to optimize cost management in multi-cloud is not out of mind.

4 Conclusion

In conclusion, multi-cloud computing has several challenges and benefits, challenges such as: workload, data, traffic portability, and vendor lock-in. On the other hand, multi-cloud computing has advantages such as effective service management, cost-efficiency, adaptability, and resource availability. The trend toward using multi-cloud computing and adopting management tools indicates the recognition of the need for efficient multi-cloud management. Prominent solutions include multi-cloud management tools and platforms. For example, DevOps tools, package management, cost optimization tools, cybersecurity measures, and container technologies. Future trends involve the integration of AI

and ML in multi-cloud management, enhanced security, and optimized cost management. Overall, multi-cloud computing provides a compelling choice for enterprises seeking resource management and access to diverse cloud services.

References

[1] Adel Abusitta et al. "A deep learning approach for proactive multi-cloud cooperative intrusion detection system". In: *Future Generation Computer Systems* 98 (Sept. 2019), pp. 308–318. DOI: 10.1016/j.future.2019.03.043.

[2] Paul Alpar and Ariana Polyviou. "Management of Multi-cloud Computing". In: *Global Sourcing of Digital Services: Micro and Macro Perspectives*. Ed. by Ilan Oshri, Julia Kotlarsky, and Leslie P. Willcocks. Cham: Springer International Publishing, 2017, pp. 124–137. ISBN: 978-3-319-70305-3.

[3] Yi-jie Bian, Lu Xie, and Jing-qi Li. "Research on influencing factors of artificial intelligence multi-cloud scheduling applied talent training based on DEMATEL-TAISM". In: *Journal of Cloud Computing* 11 (Aug. 2022). DOI: 10.1186/s13677-022-00315-4.

[4] Jens-Matthias Bohli et al. "Security and Privacy-Enhancing Multicloud Architectures". In: *IEEE Transactions on Dependable and Secure Computing* 10.4 (2013), pp. 212–224. DOI: 10.1109/TDSC.2013.6.

[5] FADDOM. *The Top 10 Multi-Cloud Solutions for 2023*. https://faddom.com/multi-cloud-solutions/. Mar. 2023.

[6] Flexera. *State of the Cloud 2022*. https://path.flexera.com/cm/report-state-of-the-cloud-2022. 2022.

[7] HashiCorp. *What is Multi-Cloud & Why are Companies Adopting it?* Video. YouTube, Aug. 2019. URL: https://www.youtube.com/watch?v=1Xnfn8p5m4Q.

[8] J. Hong et al. "An Overview of Multi-Cloud Computing". In: *Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 33rd International Conference on Advanced Information Networking and Applications (WAINA-2019)*. Vol. 33. Springer International Publishing, 2019, pp. 1055–1068.

[9] Hamza Ali Imran et al. "Multi-Cloud: A Comprehensive Review". In: *2020 IEEE 23rd International Multitopic Conference (INMIC)*. 2020, pp. 1–5. DOI: 10.1109/INMIC50486.2020.9318176.

[10] Kyriakos Kritikos, Pawel Skrzypek, and Feroz Zahid. "Are Cloud Platforms Ready for Multi-cloud?" In: Mar. 2020, pp. 56–73. ISBN: 978-3-030-44768-7. DOI: 10.1007/978-3-030-44769-4_5.

[11] LTIMindtree. *5 multi-cloud trends to look out for in 2023*. Article. LTIMindtree, 2023. URL: %7Bhttps://www.ltimindtree.com/wp-content/uploads/2023/01/5-multi-cloud-trends-to-look-out-for-in-2023-Article.pdf?utm_source=organic&utm_medium=whitepaper&utm_content=600000511762199&utm_campaign=ltimindtree+-+cloud+services%7D.

[12] Beniamino Di Martino, Antonio Esposito, and Ernesto Damiani. "Towards AI-Powered Multiple Cloud Management". In: *IEEE Internet Computing* 23.1 (2019), pp. 64–71. DOI: 10.1109/MIC.2018.2883839.

[13] P. Mell and T. Grance. "The NIST Definition of Cloud Computing". In: (2011).

[14] *Multi-cloud vs. Hybrid Cloud: What's the Difference?* https://www.cloudflare.com/learning/cloud/multicloud-vs-hybrid-cloud/.

[15] Justice Opara-Martins, R. Sahandi, and Feng Tian. "Critical Review of Vendor Lock-in and its Impact on Adoption of Cloud Computing". In: Nov. 2014. DOI: 10.1109/i-Society.2014.7009018.

[16] Dana Petcu. "Multi-Cloud: expectations and current approaches". In: *MultiCloud '13*. 2013.

[17] L. Qian et al. "Cloud Computing: An Overview". In: *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings*. Vol. 1. Springer Berlin Heidelberg, 2009, pp. 626–631.

[18] Pethuru Raj and Anupama Raman. In: *Software-Defined Cloud Centers: Operational and Management Technologies and Tools*. Cham: Springer International Publishing, 2018, pp. 185–218. ISBN: 978-3-319-78637-7. DOI: 10.1007/978-3-319-78637-7_9. URL: https://doi.org/10.1007/978-3-319-78637-7_9.

[19] Tiago Rosado and Jorge Bernardino. "An Overview of OpenStack Architecture". In: *Proceedings of the 18th International Database Engineering & Applications Symposium*. July 2014, pp. 366–367.

[20] Tara Salman et al. "Machine Learning for Anomaly Detection and Categorization in Multi-Cloud Environments". In: *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. 2017, pp. 97–103. DOI: 10.1109/CSCloud.2017.15.

[21] Faiza Samreen, Gordon Blair, and Matthew Rowe. "Adaptive decision making in multi-cloud management". In: Dec. 2014, pp. 1–6. DOI: 10.1145/2676662.2676676.

[22] Paul Schmidt. "Literature Review on Multi Cloud Management". In: *Seminar IT-Management in the Digital Age (Winter 2022)*. FH Wedel. Wedel, Germany, 2022.

[23] Statista. *Amazon Web Services: Quarterly Revenue 2014-2022*. https://www.statista.com/statistics/250520/forecast-of-amazon-web-services-revenue. Feb. 2023.

[24] Orazio Tomarchio, Domenico Calcaterra, and Giuseppe Di Modica. "Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks". In: *Journal of Cloud Computing* 9 (Sept. 2020), p. 49. DOI: 10.1186/s13677-020-00194-7.

[25] VMware. *What is Multi-Cloud Management? | VMware Glossary*. Website. Dec. 2022. URL: %7Bhttps://www.vmware.com/topics/glossary/content/multi-cloud-management.html%7D.

[26] *What is Multicloud?* https://www.redhat.com/en/topics/cloud-computing/what-is-multicloud. Accessed on: [20 May 2023].

[27] *What is vendor lock-in? | Vendor lock-in and cloud computing*. Webpage. Cloudflare, n.d. URL: https://www.cloudflare.com/learning/cloud/what-is-vendor-lock-in.

[28] T. B. Winans and J. S. Brown. *Cloud Computing: A Collection of Working Papers*. Deloitte LLC, 2009.

Business Models in the Cloud: A High Level Framework for Selecting Cloud-Based Data Solutions

Xander Akiko Snelder
11598727
University of Amsterdam
Amsterdam, The Netherlands
xander.snelder2@student.uva.nl

Thomas Webbers
13508741
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
t.w.b.webbers@student.vu.nl

Yixin Hu
14751135
University of Amsterdam
Amsterdam, The Netherlands
yixin.hu@student.uva.nl

Abstract

This essay provides a comprehensive exploration of cloud computing, focusing on its emergence, service models, and deployment models. It also investigates the crucial factors in selecting an optimal cloud service solution, considering different business needs and data types. A comparative analysis of Cloud Service Providers (CSPs) is carried out, evaluating aspects like reputation, reliability, security standards, support services, and compliance. The paper further delves into the process of optimizing and implementing cloud solutions, outlining steps for selecting an optimal plan, evaluating benefits, pricing, flexibility, and Total Cost of Ownership (TCO). Implementation steps such as migration, performance testing, reliability, and training testing are discussed, and the essay concludes with a section on monitoring and adjusting for performance and technology trends.

1 Introduction

In recent years, cloud computing has revolutionized the IT industry by providing an alternative to traditional on-premises computing. By 2021, the amount of data stored in the public cloud has surpassed both consumer devices and traditional data centers (see Figure 4 and Figure 5). Alongside this, there has also been an increase in Cloud Service Providers (CSP), which makes it challenging for businesses to choose an optimal cloud service. This literature study aims to provide a high-level framework which can help guide businesses in making this crucial decision. The scope of this study is specifically on cloud-based data solutions from three of the biggest cloud service providers: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). This paper aims to answers the following reseach question:

"*What are the key factors that influence the choice of cloud-based data solutions among businesses?*"

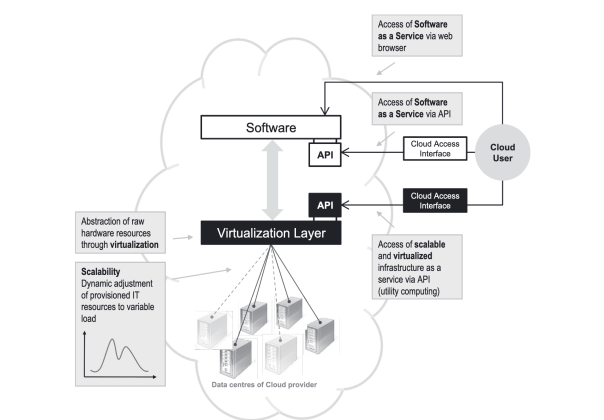


Figure 1: Defining features of Cloud Computing [21]

1.1 Emergence of Cloud Computing

Prior to the rise of cloud computing, a traditional on-premise computing infrastructure was the norm for the majority of organizations. On-premise computing refers to an internally hosted infrastructure. This means organizations are solely responsible for the design, implementation, and maintenance of their computers and servers [16]. Although this on-premise approach allows organizations to have total control over their hardware, software, and data, the authors of Peng et al. [16] mentioned that this approach carries drawbacks. Two of these are: significant upfront costs and the need for time-consuming upgrades. In contrast, cloud computing technologies offer similar infrastructure but instead, from a third-party hosted as a service over the internet. This provides a solution of the above mentioned disadvantages, through a diminishing need for initial hardware investments. Additionally this also reduces expenses and internal risks associated with system maintenance and upgrades [16].

There are many definitions for the term cloud computing, Stanoevska-Slabev et al. [21] summarize these definitions as a new computing paradigm that provides on-demand infrastructure, resources and applications. Cloud computing is characterized by its pay-per-use business model, and its main features are virtualization and on demand dynamic scalability (see Figure 1). "*Cloud Computing abstracts from the underlying hardware and system software through virtualization. The virtualized resources are provided through a defined abstracting interface (an Application Programming Interface (API) or a service). Thus, at the raw hardware level, resources can be added or withdrawn according to demand posted through the interface, while the interface to the user is not changing. This architecture enables scalability and flexibility on the physical layer of a Cloud without impact on the interface to the end user.*" [21]. In conclusion, businesses do not need to purchase, own, and maintain physical data centers and servers anymore. They can purchase resources like computing power, storage, and databases as a service from a CSP.

1.2 Service Models

Zhang et al. [27] describes three main categories of cloud computing services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). They all aim to provide the benefits of the cloud but do this in different ways. Each category delivers an increasing level of abstraction from the underlying infrastructure. All service models allow businesses to focus more on their strategic and operational tasks, and worry less about the management of the IT infrastructure. Below are the three categories as defined by Zhang et al.[27]:

1. Infrastructure as a Service (IaaS): This service model allows the provisioning of infrastructure resources over the internet on an on-demand basis, typically in the form of Virtual Machines (VMs). This eliminates the need for businesses to invest in setting up and maintaining physical servers, storage, and network infrastructure.
2. Platform as a Service (PaaS): This service model allows the provisioning of platform layer resources over the internet that encompass support for operating systems and software

development frameworks. Developers can build, test, deploy, and maintain applications without the complexities of setting up and managing the underlying infrastructure.

3. Software as a Service (SaaS): This service model provisions on-demand applications over the internet. A third-party provider hosts applications which users can access over the internet without the complications of installation and maintenance.

1.3 Deployment Models

A deployment model is a specification of how cloud computing resources are managed and who can access them. Each model has its own strengths and weaknesses, and is suitable for different needs. The optimal model typically depends on the specific requirements of a business. Mell and Grance [12] have described four deployment models:

1. Private cloud: This cloud infrastructure is exclusively provided for the sole utilization of a single organization containing multiple business units. The ownership, management, and operations are hosted on a private network and are maintained by the organization itself, an external third party, or a combination of both. The physical location of the private cloud can be either on-site (on-premises) or off-site.
2. Community cloud: This cloud infrastructure refers to a collaborative system utilized by a particular community of users that share a common interest. The ownership, management, and operations are maintained by one or more of the organizations in the community, an external third party, or a combination of both. Just like the private cloud, the community cloud can exist either on or off-premises.
3. Public cloud: This cloud infrastructure is openly available for the use of the general public. The ownership, management, and operations can be a diverse range of entities including businesses, academic institutions, government bodies, or a combination of them. The public cloud is physically located at the cloud provider.
4. Hybrid cloud: This cloud infrastructure is a combination of two or more previously mentioned distinct cloud deployment models (private, community, public). Each deployment model retains their distinct identity, but is interconnected through standardized or proprietary technology. This interconnection facilitates data and application portability. Hybrid clouds can offer the security of private clouds, the collaborative potential of community clouds, and the scalability of public clouds.

2 Defining the Business Needs

The first and possibly the most critical step in choosing the optimal cloud service solution is defining a business they needs. Businesses should consider several key factors, for example, a business dealing with mainly ordinal data might use different analytical tools than a business primarily handling interval data. Different data types may require distinct data storage and management systems, which in turn can affect costs, efficiency, and performance. [19]

2.1 Storage of Different Data Types

A business should understand the type of data it processes before selecting the optimal cloud service data solution. *"Due to their different characteristics, each data type requires a specialized storing system, as inappropriate storing reduces performance, robustness, flexibility, and scalability. Hence, it is important to identify a sophisticated strategy for storing and synchrosizing different types of data structures in a way they provide the best mix of the previously mentioned properties."* [19]

2.1.1 Qualitative Data

According to Heiss [9], data can generally be categorized as nominal, ordinal, interval, and ratio. Both nominal and ordinal data are referred to as categorical or qualitative data. Examples of qualitative data are raw textual data from interviews, online content, or documents. This kind of data needs minimal structuring to be analyzed. The main difference between nominal and ordinal data is that ordinal data has an inherent order or ranking, and nominal data does not.

2.1.2 Quantitative Data

Interval and ratio data are both defined as numerical or quantitative data [9]. Quantitative data is characterized by a high degree of standardization, and can be obtained through close-ended questions. This type of data needs to be ranked before analyzed. The main difference between ratio and interval data is that interval data has no natural zero point, thus interval data will have negative values and ratio will not.

2.2 Big Data Challenges

Although there is still disagreement among data specialists about the precise definition of big data, in particular about the number of "V's" (this will be elaborated upon below) that should be considered, Lake and Crowther [11] characterize big data as volume, velocity, variety, and veracity. These big data characteristics require high-demanding storage, processing, and analysis systems [4]. This need is further emphasized by the continuing global surge in data generation. By understanding their own data characteristics, a business can better decide which CSP they need when deploying a cloud-based data solution. For instance, high data volume requires robust and scalable storage solutions, while high-velocity data demands capabilities for real-time processing and analysis [4].

2.2.1 Volume

Volume is the most straightforward aspect of big data. The storage of data - a crucial requirement of any data-driven business - is an important consideration when dealing with large volumes of data. Businesses need to understand their current storage requirements in terms of volume and scalability. These two aspects may have a big impact on the ability to efficiently manage and utilize data [4]. The International Data Corporation (IDC) predicts that the Global Datasphere will grow from approximately 33 Zettabytes in 2018, to 175 Zettabytes by 2025 (see Figure 5). Furthermore, 49% of the world's stored data will reside in public cloud environments, and 75% of the global population will interact with data on a daily basis [18]. As stated by Lake and Crowther[11], volume will bring the most significant challenge for data professionals in the upcoming future.

2.2.2 Velocity

Not only the volume of data will pose significant challenges, but also the velocity will cause substantial complexities. The velocity of big data refers to the speed at which data is instantly being generated, processed, and analyzed [11]. In 2020, approximately 5% of the global data sphere is represented by real-time data. By 2025, this number will increase to 30% (see Figure 7). Furthermore, the average data interaction per day per person connected to the internet will increase from 1,426 in 2022, to 4,900 by 2025 (see Figure 8).

2.2.3 Variety

The variety of big data refers to the different types of data that businesses have to deal with. In general, three types of data can be considered: structured data (e.g., clean and processed data in databases), semi-structured data (e.g., JSON and XML files), and unstructured data (e.g., emails and social media content) [5]. The majority of data is represented by unstructured data, approximately 80% of organizational data is unstructured [19]. It requires processing power and skilled employees to structure and clean data. Each type of data may require different storage, processing, and analytical capabilities. This data variety influences which CSP is most suitable.

2.2.4 Veracity

The veracity of big data is characterized as the quality and trustworthiness of data [11]. It is a crucial factor in ensuring the quality of data analysis and the efficacy of machine learning models. If the quality of data is compromised, it inevitably undermines the reliability of data analytics and machine learning models - a classic case of 'garbage in, garbage out'. When choosing a cloud-based data solution, businesses should evaluate the tools and services provided by the CSP for maintaining data quality assurance and data governance.

2.3 Processing Power and Storage

Businesses should perform a thorough analysis of the current state of data-related operations and usage within the organization. Some businesses will suffice with simple data processing and analytics, meanwhile, others needs more advanced data solutions. The use of real-time data analytics often requires more immediate processing power than batch processing [4]. Furthermore, complex machine learning models with large datasets require a significant amount of computing power, and demand optimized algorithms for training and deployment [4]. Understanding how a business utilizes its data and the amount of processing power required, is an important aspect in finding the optimal cloud-based data solution.

Additionally, businesses should anticipate their potential to upscale, a critical element that might be overlooked upon. Expanding or growing a business might influence the demand for resources in the future, for example, the requirements related to storage power and processing power. Businesses should choose a CSP that offers the scalability to accommodate future growth or expansion.

2.3.1 Data Processing Techniques

Stream processing is defined as data being processed immediately when it arrives, meanwhile, batch processing is when data is collected over some period of time and then processed at once [4]. Batch processing is more efficient since it handles large amounts of data at the same time, which is particularly useful when real-time data is not required. Examples of batch processing use cases are Extract Transform Load (ETL) jobs and the training of machine learning models with large amounts of training data. On the other hand, stream processing involves the continuous input and output of data. Stream processing is particuraly useful for applications that demand timely or immediate insights or responses [4]. Examples of use cases of stream processing are fraud detection in banking transactions, live monitoring of system logs in cybersecurity, and monitoring of Internet of Things (IoT) devices.

2.3.2 Example of Data Processing Techniques

In some applications, both batch processing and stream processing can be utilized. For instance, a financial organizations that deploys a machine learning model to detect fraud detection in credit card transactions [4]. Initially, batch processing is performed to train the model on a historical dataset of transactions. Once the model is trained, it recognizes non-fraudulent and fraudulent patterns. Stream processing can be used to deploy the model in a live environment that evaluates the transactions in real-time. The model makes predictions on whether a transaction is fraudulent or not using live data. This allows the financial organization to take immediate action when a potential fraud is detected, and leverages both the efficiency of batch processing and the responsiveness of stream processing.

3 Comparison of Cloud Service Providers

Business operations encompass a myriad of practical requirements, thus necessitating a comprehensive, multi-dimensional analysis when determining the optimal cloud service provider. This chapter serves to methodically compare the leading service providers in the market - Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) - based on a variety of critical aspects.

Although additional factors, such as the geographical location of data centers, may warrant consideration, the scope of this analysis is primarily directed towards the most significant and relevant elements influencing the selection of cloud-based data solutions.

The focus on AWS, Azure, and GCP is motivated by their predominant positions in the market and their extensive service portfolios. This analytical approach provides a balanced and in-depth perspective that aids businesses in making an informed decision consistent with their operational needs and strategic objectives.

3.1 Reputation

Reputation, reflecting public sentiment and a multitude of user experiences, is fundamental as it serves as a proxy for the commitment to quality by the service provider. Among the three providers,

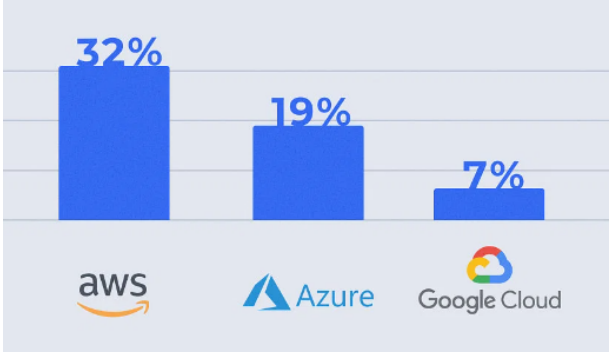


Figure 2: Top 3 Market Share of Cloud Service Providers. [1]

Amazon Web Services (AWS) boasts the longest operational history and arguably the most esteemed reputation in the industry. As the preferred choice of many leading global firms, AWS commands a substantial share of the cloud industry, at least as of my knowledge cutoff in September 2021. Although Microsoft's service, Azure, entered the market later than Amazon Web Services (AWS), it has quickly increased its market share, particularly among larger companies that already utilize other Microsoft products. Despite being the newest of the three major providers, Google Cloud Platform is developing a solid reputation. Because of Google's long-standing history in data processing, many firms have confidence in it.

3.2 Reliability

Reliability, signifying a provider's ability to uphold continuous and unwavering service, is an essential factor as it ensures operational continuity and decrease potential losses for businesses. AWS, Azure, and Google Cloud have distinguished themselves as reliable providers in the cloud service landscape. AWS has a proven track record of providing extremely dependable services, which is supported by its remarkable resume [2]. AWS guarantees an uptime of 99.99% through its Service Level Agreement (SLA). Similar to AWS, Azure offers a comparable SLA that ensures dependability and availability thanks to its resilient architecture[13]. The infrastructure of Google Cloud is also strong, and it offers a SLA that is competitive with both AWS and Azure, highlighting its dedication to providing trustworthy services[8].

3.3 Security Standards

The ability of service providers to protect sensitive data from potential threats is fundamental, particularly for businesses dealing with sensitive or confidential data. AWS provides a comprehensive array of global, cloud-based security services, including threat detection, data encryption both at rest and in transit, and DDoS mitigation. It adheres to a shared responsibility model, where AWS ensures cloud security, while the client manages security in the cloud [2]. Microsoft Azure provides a comparable set of security services and follows a shared responsibility paradigm. Furthermore, Microsoft has a lengthy history in enterprise security, making it a reliable choice for many businesses[13]. GCP also provides a robust set of security measures and adheres to the same shared responsibility paradigm. GCP is deemed trustworthy and secure due to Google's vast experience with internet security[8].

3.4 Support Services

The provision of sufficient support by cloud service providers is paramount to swiftly address potential issues, thereby minimizing downtime and limiting potential impacts on operational efficiency.

AWS offers an array of support plans, extending from complimentary basic support to more premium plans such as Developer, Business, and Enterprise. These plans present varying levels of service to cater to diverse business needs.

Azure mirrors this approach, offering four distinct support levels: Basic, Developer, Standard, and Professional Direct, each characterized by different service levels, response times, and availability of support engineers.

Similarly, Google Cloud Platform provides a spectrum of support levels, ranging from complimentary to premium. Higher tiers provide enhanced access to technical support and expedited response times, ensuring businesses receive the level of support commensurate with their needs.

3.5 Compliance

This perspective is critical for businesses operating in regulated sectors or with sensitive data, ensuring they meet their legal obligations. Privacy is an important aspect when choosing a cloud-based service. Different industries have specific data privacy requirements and regulations. This is particularly significant when business are dealing with highly sensitive data, which requires different levels of protection and privacy mechanisms. [20]. AWS is a suitable option for companies in highly regulated industries due to its extensive list of compliance certifications, which includes GDPR, HIPAA, FedRAMP, SOC 1/2/3, and ISO 27001. Azure also has a long list of compliance certifications and is well renowned for being a top option for many companies in regulated industries, especially due to its potent hybrid cloud configuration features. Similar to AWS and Azure, GCP offers a long list of compliance certifications and has made progress toward being more accommodating to regulated businesses.

4 Optimizing and Implementing Solutions

Once businesses have defined their requirements and compared potential cloud service providers (CSPs), the next crucial step is to select the right cloud service plan. This decision should be guided by the plan's features, pricing model, flexibility, and total cost of ownership (Marston, Li, Bandyopadhyay, Zhang, & Ghalsasi, 2011).

4.1 Selecting and Evaluating the Optimal Plan

4.1.1 Benefits of cloud computing

Cloud computing offers a pletora of new features. The book: "Cloud Computing Fundamentals"[6] identifies the following:

1. Scalability: Cloud computing allows for resources on demand and due to the new advances in micro service architecture horizontal scaling also allows for scalability for individual services.
2. User-centric interface: The distributed and isolated nature of cloud interfaces means that they can be accessed regardless of location. Furthermore they are ussually reachable via common tools such a web browsers.
3. Guaranteed Quality of Service: Possibly the most important reason for its inception, due to its monitoring features coupled with its distributed nature, cloud computing can provide QoS guarantees in CPU performance, bandwidth and memory.
4. Autonomous sytem: Cloud computing piggybacks on the exiting internet routing protocols to automaitcally route a service correctly to a user. Additionally data from the services can be downloaded to a single machine for ease of use.
5. Pricing: Cloud computing does not require intital hardware costs or even maintenance allowing for lower startup capital

4.1.2 Cloud computing Price

Pricing is an important aspect to consider when choosing a cloud computing service.The book: "Cloud Computing Fundamentals"[6] has this nice table giving an overview of pricing:

Resource	UNIT	Amazon	Google	Microsoft
Stored data	GB per month	$0.10	$0.15	$0.15
Storage transaction	Per 10 K requests	$0.10		$0.10
Outgoing bandwidth	GB	$0.10 – $0.17	$0.12	$0.15
Incoming bandwidth	GB	$0.10	$0.10	$0.10
Compute time	Instance Hours	$0.10 – $1.20	$0.10	$0.12

Figure 3: Price comparison for CSP
[6]

4.1.3 Flexibility

Flexibility refers to the ability of the service plan to adapt to changes in the business's needs and demands. Zhang et al. [26] seems to suggest that due to the modular nature of cloud services they are inherently flexible when it comes to a business needs. Therefore flexibility does not need to be considered throughly when slecting a CSP.

4.1.4 TCO (Total Cost of Ownership)

The cost of ownership of a cloud computing services goes beyond upfront costs. Walterbusch et al.[23] found that decisions about cloud computing services are usually adhoc. Furthermore, due to factors such as vendor lock-in and bandwidth cost it is not straight forward to determine. They divided this cost into four categories:

1. Initiation: An analysis of current and desired IT infrastructure and services, choice of cloud type and a definition of service types and programming environment (language and tools
2. Evaluation: The search process of finding all the aspects of initiation as well as a complete evaluation of the capabilities of the CSP and their reputation. The reported price is also taken into consideration, but the price that the CSP provides is often found to be severely underestimated.
3. Transition: Implementation, configuration, integration and migration of the service, this also includes access authorisation and merging into existing IT infrastructure.
4. Operation: Support, initial training, ongoing training, maintenance and modification, system failure and recovery and back sourcing.

Walterbusch et al.[23] also provides equations for all these individual components per cloud and service type but that goes far outside the scope of this paper.

4.2 Implementation and Testing

4.2.1 Migration

Migration: Migration is the process of transferring data, applications, and processes from the current system to the cloud environment. Rai et al. [17] did an extensive study on cloud migration and found that cloud migration nearly always lacks dedicated support and tools. CM-tools do exist but often have a specific focus and are infrequently used [17]. Additionally they propose a five step methodology to migration.

1. Feasibility study
2. Requirement analysis and migration planning
3. Migration execution
4. Testing and migration validation
5. Monitoring and maintenance

Another important aspect to consider when migrating is vendor lock-in[15]. Opera-Martins et al. found that one of the biggest barriers to cloud adoption is the risk vendor lock-in presents.

Furthermore the cost of changing providers is often under appraised until the moment of migration, incurring more hidden costs. There is currently work underway to propose cloud-specific standards, but because such standards affect the implementation level it is difficult to come to an consensus.[15] Currently the best way to prevent vendor lock-in is a well thought out analysis that is case specific for every application.

4.2.2 Performance testing

Performance Testing is the process of evaluating the efficiency and speed of the cloud services under different load conditions. Gilliam et al. [7] has done extensive research on cloud benchmarks. First they looked at services that provided cloud comparison services. These were: CloudHarmony, Cloudsleuth and OpenBenchmarking.org. A business looking to do performance testing and/or choose a CSP based on performance would do well to consult these resources. Additionally Gilliam et al. [7] has also created and performed their own benchmarks, and offers these as a web portal when contacted [7], which is an extra step that a business can perform.

4.2.3 Reliability Testing

"Reliability in the cloud refers to the probability that the cloud delivers the services it is designed for" [25]. As much as businesses would want this to be a guarantee the reality is that reliability is difficult to achieve. Furthermore, none of the existing cloud services has achieved complete reliability. [25] What can be achieved however is a robust stress testing system. Chaos Monkey [14] and its successors belong to suite of open source tools called "Simian Army" [25]. Simian Army tests for resiliency of cloud operations, covers reliability, security and recoverability. The basic idea behind these tools is that they kill random micro services and VM's and then check if the system still works. It turns out that for software as well as people that the only way to avoid failure is to fail over and over. This way of testing has become the de-facto standard for reliability testing and should definitely be performed by any business deploying full scale software solutions via the cloud.

4.2.4 Training

As should now be obvious cloud computing technologies are quite complex. As is the case with any complex system employees require extensive training. Global Digital Infrastructure did a survey of 1212 companies and found that 70% of the respondents use SaaS technologies. Despite its wide adoption employees have insufficient knowledge about the cloud and its benefits.[3] Training has not kept up with the technological advancements. [10] Kahle et al. proposes three ways in which employees and the industry can be trained:

1. E-learning: online traineeships like the once offered by Kubernetes and AWS
2. Employee training: specialised in-house employee trading given by hired professionals
3. Student workshops: Due to the wide adoption of cloud computing, more and more workshops and talks are given by students which are open for business employees as well

No significant differences were found in training needs for different CSP's. However, AWS and larger CSP's do offer cloud traineeships and certificates[22], this would be a good first step for a business looking to train their employees.

4.3 Monitoring and Adjusting

4.3.1 Performance monitoring

The rapid change in scale composition neccesitates for cloud computing neccesitates sophisticated performance monitoring. [24] Ward et al. [24] does an extensive study of both cloud and native systems monitoring. They find that presently there is no universal monitoring strategy to select such tools. Furthermore "most monitoring strategies are a patchwork of several monitoring tools each which provide different functionality." [24]. The most important creterions in such systems include: health and network monitoring, metric gathering and log/event processing. Lastly it should be mentioned that both Amazon and Google have made strides in this regard. Amazon has a service known as Amazon Cloudwatch and Google has defined a new type of engineer namely a SRE (Site Reliability Engineer). [24]

4.3.2 Technology and Trend Monitoring

This involves staying abreast of the latest trends and developments in the cloud computing industry to identify potential opportunities for improving the cloud service plan. Unfortunately this information does not seem to be available in any literary work. It is also not clear from the CSP's their own metrics and data. Monitoring something like this would require expert knowledge of the emerging technologies and wether they are implemented within the various CSP.

5 Discussion

This research has presented a birds eye view of the factors that influence a business their choice of cloud solution. As such the first point of discussion is quantity over quality. Rather than focus on a selection of the most important factors in detail the conscious choice was made to offer a complete but somewhat shallow overview. As such the complete computations from Walterbusch et all[23] are not included in this research despite these being very useful for anyone selecting a cloud computing solution. Another point of contention in this research was the absence of information for three topics, namely: Usability testing, System configuration and access control and lastly Support monitoring. It would have been good to know which cloud solution was considered most user friendly, easy to setup and also which had the best user support, regrettably no research could be found on this topics. Perhaps this is an avenue of research for another group. Additionally, the information on training needed to use the different cloud technologies was quite lacking, information that was considered quite vital to have when making a choice for a cloud solution. Lastly, it should be mentioned that a significant part of the corpus of this literature study uses information dating back several years which, considering the speed of development in the cloud eco system, might make some of it dated.

6 Conclusion

The emergence, service models, and deployment models of cloud computing are all thoroughly examined in this essay. Additionally, it looks into the critical criteria for deciding on the best cloud service solution while taking into account various business requirements and data types. It compares different Cloud Service Providers (CSPs) and assesses their reputation, dependability, security requirements, customer assistance, and compliance. The process of designing and implementing cloud solutions is further explored in the paper, which also outlines how to choose the best strategy and assess its pros, cons, price, flexibility, and total cost of ownership (TCO). The essay concludes with a section on monitoring and making adjustments for performance and technological developments. Implementation stages such as migration, performance testing, reliability testing, and training testing are described.

References

[1] Accenture - New insights. Tangible outcomes. New Applied Now, 2021. Accessed: 2023-05-25.

[2] Amazon Web Services. AWS Documentation. `https://aws.amazon.com/documentation/`, 2023.

[3] Inaki Bildosola, Rosa Río-Belver, Ernesto Cilleruelo, and Gaizka Garechana. Design and implementation of a cloud computing adoption decision tool: Generating a cloud road. *PloS one*, 10(7):e0134563, 2015.

[4] C. P. Chen and C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.

[5] A. C. Eberendu. Unstructured data: an overview of the data of big data. *International Journal of Computer Trends and Technology*, 38(1):46–50, 2016.

[6] Borko Furht. Cloud computing fundamentals. *Handbook of cloud computing*, pages 3–19, 2010.

[7] Lee Gillam, Bin Li, John O'Loughlin, and Anuz Pratap Singh Tomar. Fair benchmarking for cloud computing systems. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):1–45, 2013.

[8] Google Cloud. Google Cloud Documentation. `https://cloud.google.com/docs`, 2023.

[9] R. Heiss. Data, types of. *The International Encyclopedia of Communication Research Methods*, pages 1–6, 2017.

[10] Lisa Kahle-Piasecki, Matthew E Ritzman, and Doug Ellingson. Up in the cloud: Managers, employees, and security training for cloud computing to avert cyber threats. *American Journal of Management*, 17(7):58–63, 2017.

[11] P. Lake and P. A. Crowther. *Concise Guide to Databases*. 2013.

[12] Peter Mell and Timothy Grance. The nist definition of cloud computing. *National Institute of Standards and Technology (NIST)*, page 3, 2011.

[13] Microsoft. Azure Documentation. `https://docs.microsoft.com/en-us/azure/`, 2023.

[14] SAVED YOUR NETFLIX. Chaos engineering saved your netflix. *system*, 1(4).

[15] Justice Opara-Martins, Reza Sahandi, and Feng Tian. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5:1–18, 2016.

[16] George Peng and Cipriano Gala. Cloud erp: A new dilemma to modern organisations? *Journal of Computer Information Systems*, 54(3):22–30, 2014.

[17] Rashmi Rai, Gadadhar Sahoo, and Shabana Mehfuz. Exploring the factors influencing the cloud computing adoption: a systematic study on cloud migration. *SpringerPlus*, 4:1–12, 2015.

[18] D. Reinsel, J. Gantz, and J. Rydning. The digitization of the world: From edge to core. Technical report, International Data Corporation (IDC), 2018.

[19] R. Sint, S. Schaffert, S. Stroka, and R. Ferstl. Combining unstructured, fully structured and semi-structured information in semantic wikis. In *Fourth Workshop on Semantic Wikis–The Semantic Wiki Web 6 th European Semantic Web Conference*, Hersonissos, Crete, Greece, June 2009.

[20] D. J. Solove and P. M. Schwartz. *Information Privacy Law*. Wolters Kluwer, New York, 2003.

[21] Katarina Stanoevska-Slabeva, Tim Wozniak, and Srdjan Ristol. *Grid and Cloud Computing: A Business Perspective on Technology and Applications*, pages 47–50. 2009.

[22] Sarah Walker. Aws training, 2011.

[23] Marc Walterbusch, Benedikt Martens, and Frank Teuteberg. Evaluating cloud computing services from a total cost of ownership perspective. *Management Research Review*, 36(6):613–638, 2013.

[24] Jonathan Stuart Ward and Adam Barker. Observing the clouds: a survey and taxonomy of cloud monitoring. *Journal of Cloud Computing*, 3(1):1–30, 2014.

[25] Mazin Yousif. Cloud computing reliability—failure is an option. *IEEE Cloud Computing*, 5(3):4–5, 2018.

[26] Lin Zhang, H Guo, Fei Tao, YL Luo, and N Si. Flexible management of resource service composition in cloud manufacturing. In *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 2278–2282. IEEE, 2010.

[27] Qi Zhang, Liang-Ping Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

A Appendix

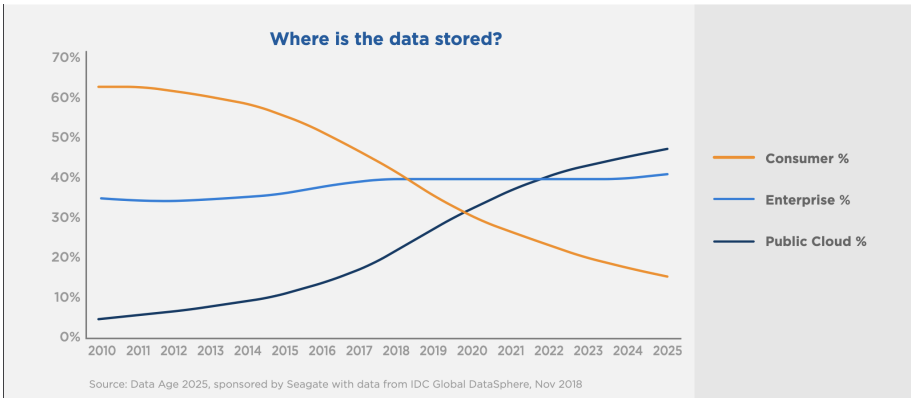


Figure 4: Where is the data stored? [18]

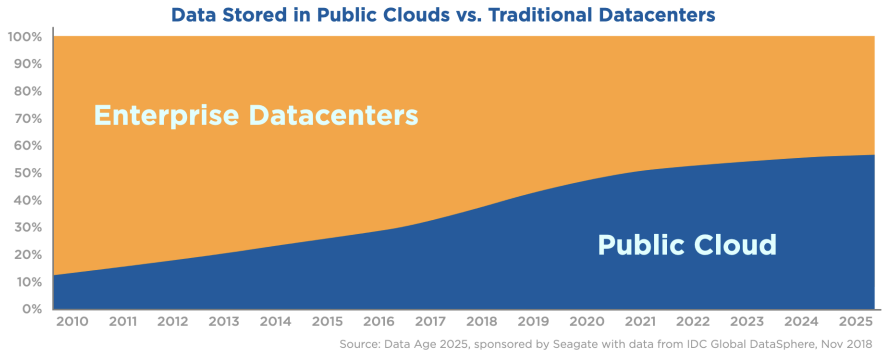


Figure 5: Data Stored in Public Clouds vs. Traditional Datacenters [18]

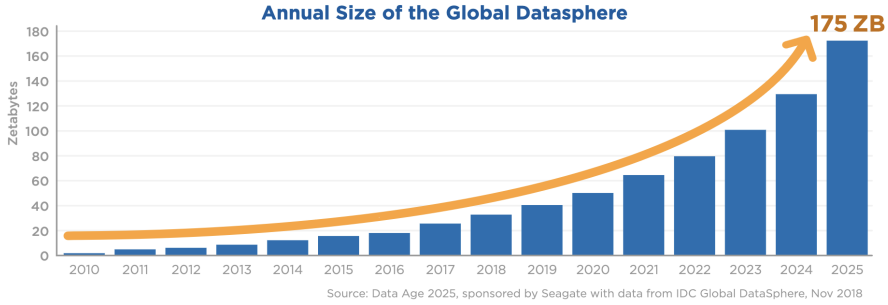


Figure 6: Annual Size of the Global Datasphere [18]

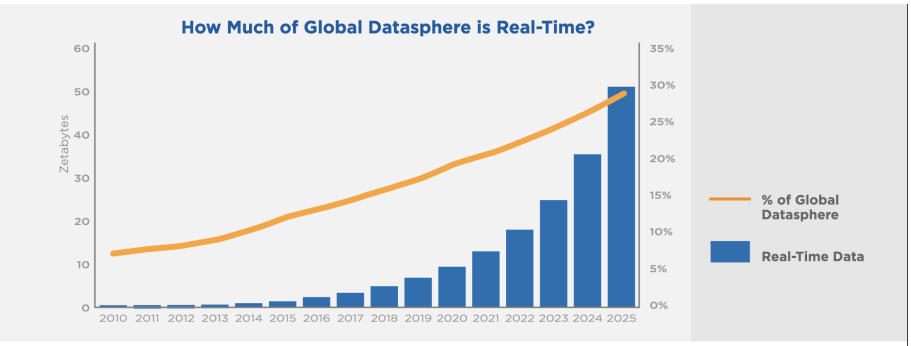


Figure 7: How Much of Global Datasphere is Real-Time? [18]

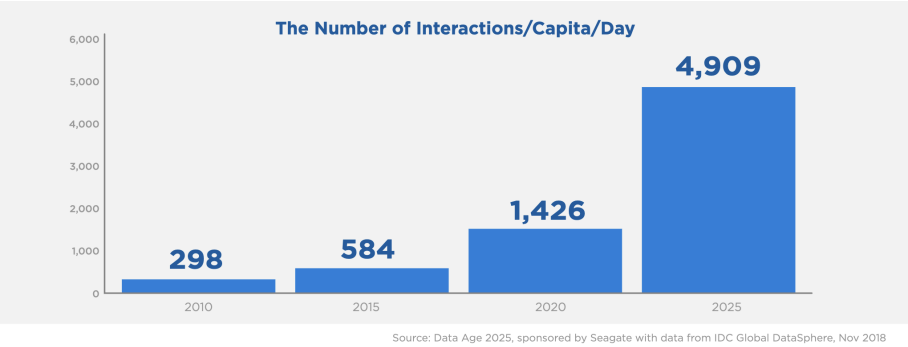


Figure 8: Data Interactions per Connected Person Per Day [18]

Performance of Serverless computing (lambda function), FaaS (functions as a service)

Ye Yuan
University of Amsterdam
ye.yuan2@student.uva.nl

Shaojie Hou
University of Amsterdam
shaojie.hou@student.uva.nl

Yuhang Zhu
University of Amsterdam
yuhang.zhu@student.uva.nl

Abstract

This paper explores the performance aspects of serverless computing. The background and benefits of serverless computing are first presented, including improved agility, scalability, and cost-effectiveness. Then the importance of performance evaluation is discussed, including understanding latency and efficiency, identifying opportunities for optimization, and reducing costs. Benchmarking and tools for performance evaluation are then presented, as well as methods and important concepts for performance modeling, such as function instances and cold starts. In the performance optimization section, methods for reducing cold start time, such as retry mechanisms, reducing the depth of the function call stack, circuit breakers, and warm-up strategies, are explored. In addition, some real-world case studies are given to show quantitative evaluation and comparison of the performance of leading serverless platforms. Finally, challenges and future directions of serverless computing are discussed, including containerization, utilization of legacy code, stateful serverless, service-level agreements, edge computing, and new application scenarios.

1 Introduction

Serverless computing represents a cloud computing paradigm that enables developers to create and execute applications without the burden of managing underlying server infrastructure. By leveraging this model, developers can solely focus on crafting code for their application's core functionality, while the cloud provider assumes responsibility for scaling, provisioning, and server resource management. The benefits of serverless computing encompass heightened agility, scalability, and cost-efficiency, as developers only pay for the actual execution time of their functions or services. Consequently, exploring and understanding the performance aspects of serverless computing becomes crucial to optimize resource utilization, meeting user expectations, and achieving cost savings.

The significance of studying serverless computing performance lies in various key factors. Firstly, with the increasing popularity of serverless computing, evaluating its performance characteristics becomes paramount for ensuring efficient resource utilization and meeting the demands of users. By understanding the latency and efficiency of serverless infrastructures, developers can adeptly create and launch applications capable of managing diverse workloads. Furthermore, evaluating performance helps identify possible obstacles, areas with room for improvement, and optimization approaches that enable developers to improve the overall performance and user satisfaction of

serverless applications. Furthermore, considering the "pay-as-you-go" structure of the serverless business model, optimizing performance directly leads to substantial cost reductions.

The objective of this article is to offer a thorough overview of the performance aspects of serverless computing. The structure encompasses multiple sections, starting with a deep dive into performance evaluation. Here, we analyze various metrics associated with serverless applications and discuss diverse approaches and methodologies for measuring and benchmarking the performance of serverless functions and services. Subsequently, we delve into performance optimization methods specifically tailored to serverless computing. This section covers a range of techniques aimed at boosting the efficiency and responsiveness of serverless applications. In addition, practical case studies are provided to exemplify instances of optimizing performance within serverless architectures. Finally, the article examines the obstacles and future paths of performance research in serverless computing, shedding light on emerging trends, areas for enhancement, and potential solutions that can further elevate the performance of serverless applications.

2 Performance Evaluation

2.1 Benchmarking

In the paper *Benchmarking Serverless Computing Platforms*, Horácio Martins and Filipe Araújo propose a benchmarking test suite and an open-source software tool to evaluate the performance of cloud serverless platforms.

2.1.1 Performance Evaluation Tests

In this study, the authors propose a suite of seven tests to evaluate the performance of serverless computing platforms. These tests aim to assess various aspects of performance, including latency, throughput, scalability, container lifecycle management, programming language impact, memory allocation, and performance under computationally intensive tasks.

For example, one of the tests, the T1 Overhead test, focuses on evaluating the overhead imposed by the serverless platform. It measures the latency of a low-computational-effort function to isolate the execution time from other factors. By sequentially invoking the function over time, the latency of each invocation is measured. The T1 Overhead test provides insights into the efficiency of the serverless platform and helps to understand the impact of the platform's infrastructure on the overall performance. This test is particularly useful in understanding the baseline performance characteristics of the platform and can serve as a benchmark for comparison with other tests and platforms.

2.1.2 Benchmarking Tool

To facilitate the performance evaluation of serverless computing platforms, the researchers developed a benchmarking tool [10]. This tool automates the execution of tests, data collection, and generation of performance plots. It offers extensibility, allowing the inclusion of additional tests, and it is platform-agnostic. The tool's architecture, depicted in Figure 1 following Simon Brown's C4 model [2], involves utilizing the open source serverless framework toolkit [5] and JMeter [1]. It employs a command-line interface and a test module, with file repositories storing the functions' code, JMeter configuration files, and test results. Python was used for implementation. By utilizing this benchmarking tool, the researchers effectively automated the performance evaluation process, enabling comprehensive assessments of various serverless computing platforms.

2.2 Performance Modelling

In this section, we delve into the work of Nima Mahmoudi and Hamzeh Khazaei, who explored the field of performance modeling in their paper *Performance Modeling of Serverless Computing Platforms*, focusing on the understanding of scheduling algorithms and function instance states. While official documentation on scheduling algorithms is scarce, previous research has shed light on these algorithms through experimental investigations [18], [9], [7], [14]. Leveraging these findings and conducting their experiments, they aim to develop a tractable and accurate performance model for modern serverless computing platforms. They introduce some concepts which are important to the performance evaluation of serverless computing platforms.

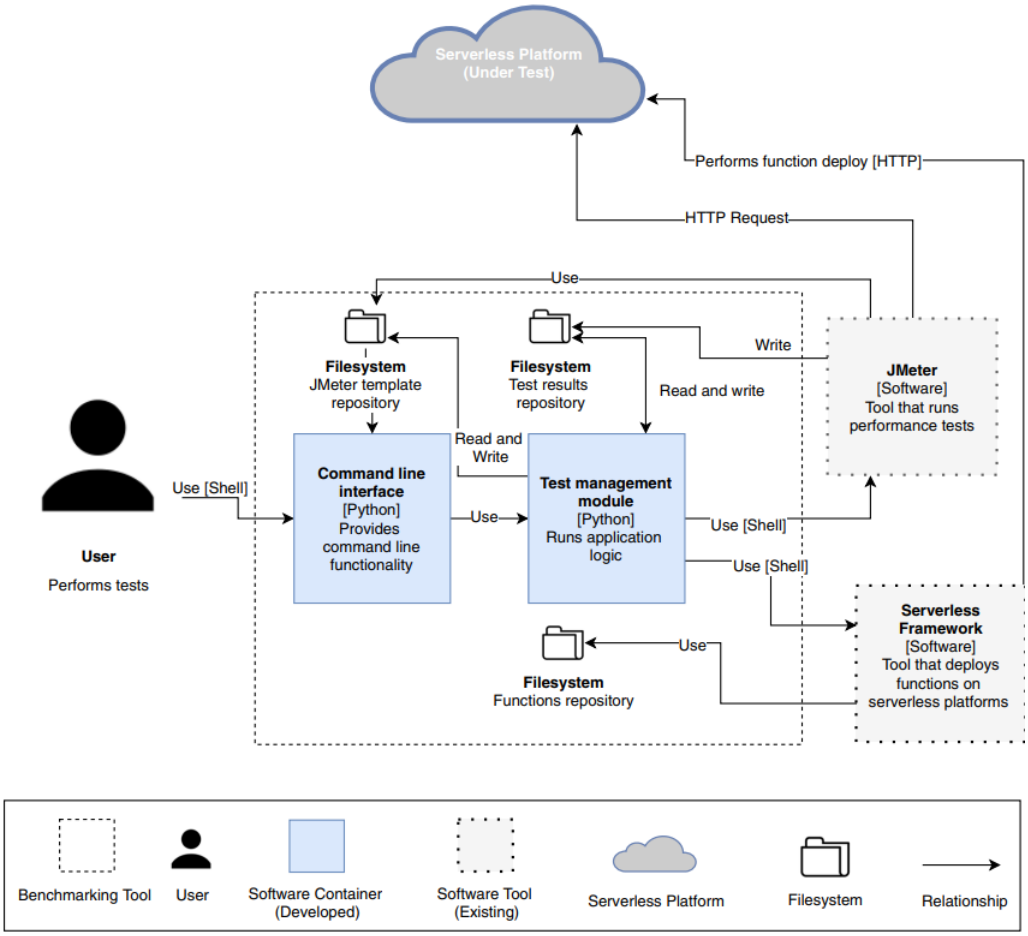


Figure 1: Container diagram of the benchmarking system[16]

2.2.1 Function Instances

In serverless computing, computation is performed in function instances. These instances are fully managed by the serverless computing platform provider and act as small servers to handle incoming requests. The research paper emphasizes the importance of understanding the different states of function instances, such as initializing, running, and idle, as well as the associated costs and billing policies.

2.2.2 Cold Start and Warm Start

In the performance evaluation of serverless computing platforms, the concept of cold/warm start is of significant importance. A cold start refers to the situation where a request triggers the launch of a new function instance, which involves various setup activities such as provisioning virtual machines or deploying functions. This results in additional overhead and increased response time for users. On the other hand, a warm start occurs when a platform already has an idle instance available and can reuse it for incoming requests without the need for launching a new instance. Warm starts are typically faster than cold starts as they avoid the setup overhead. However, cold starts can be significantly longer than warm starts, impacting application responsiveness and user experience. To address this issue, extensive research has been conducted in the serverless computing field to mitigate cold starts and improve overall performance [13], [6], [15]. Various techniques and strategies have been proposed to minimize the occurrence and impact of cold starts, aiming to enhance user satisfaction and application performance.

2.2.3 Initialization Time

The initialization time refers to the duration from when a request is received by the serverless platform until a new function instance is fully operational and ready to handle the request. It consists of two components: platform initialization time and application initialization time. The platform initialization time is the time taken by the serverless platform to prepare the function instance, while the application initialization time is the time taken by the application to perform its initialization tasks, such as establishing database connections or loading libraries. Optimizing the initialization time is essential for enhancing the performance of serverless applications and improving user experience.

2.2.4 Response Time

In the context of scale-per-request serverless computing platforms, the response time typically comprises two components: queuing time and service time. However, in these platforms, queuing time is not applicable as incoming requests do not experience any queuing delay. Serverless computing platforms exhibit linear scalability, meaning that the distribution of response time remains consistent across varying workloads. To analyze and model the response time in serverless computing platforms, delay centers [11] are employed, providing a framework for analytical evaluation. By leveraging delay centers, the response time can be quantitatively studied and optimized within serverless architectures.

3 Performance Optimization

In this section, we discuss performance optimization methods that can be implemented for serverless computing. Most of these methods revolve around minimizing the cold start and initializing time of lambda functions, as they usually play the biggest role in serverless computing performance [10].

Firstly, a retry scheme can be implemented such that callers can simply take the next free resource instead of waiting for a cold lambda to instantiate. It is important to consider the trade-off in this approach, as retrying can potentially spin up a number of lambda functions [17].

Secondly, the depth of the lambda function call stack should be taken into consideration when designing a serverless computing-backed structure. As lambda functions have cold starts, this waiting time can also accumulate with the call stack, therefore the lambda call chain should be kept short.

Thirdly, circuit breakers can be used to mitigate resource consumption when lambda functions are unresponsive or erroneous. It can be identified that certain lambda functions should not be retried because it's returning too much error or simply unresponsive, a circuit breaker can be used to route requests to another handler, presumably one that is less prone to error. This way, we can prevent starting up new lambda functions that are doomed to fail, saving resources.

Finally, a warm-up strategy can be used so that a number of lambda functions are kept warm at all times [4]. If dummy calls are made to the lambda functions periodically, we can make sure that some lambda functions are always available for use when needed. In AWS, this can be implemented with Provisioned Concurrency. With Provisioned Concurrency enabled, functions are kept ready to respond to requests, and start-up would be predictable [3].

In conclusion, performance optimization in serverless computing involves implementing various methods to improve resource availability, reduce latency, and mitigate errors. Firstly, a retry scheme can be employed to minimize waiting time by allowing callers to use the next available resource instead of waiting for a cold lambda instantiation. However, the trade-off of potential resource consumption should be carefully considered. Secondly, to reduce cold start delays, the depth of the lambda function call stack should be kept minimal, ensuring that waiting times do not accumulate with each function call. Thirdly, circuit breakers can be utilized to prevent resource wastage when unresponsive or error-prone lambda functions are encountered. By diverting requests to alternative handlers, the use of resources can be optimized. Lastly, a warm-up strategy, such as using Provisioned Concurrency in AWS, can ensure that a subset of lambda functions is kept warm at all times. Regular dummy calls can be made to these functions, maintaining their readiness and predictability. By implementing these performance optimization methods, serverless computing services can achieve improved efficiency, reduced latency, and enhanced resource management.

4 Case Studies

Jinfeng Wen, Yi Liu, et al.[19] conducted a comprehensive study on the characteristics of mainstream commodity serverless computing platforms (i.e., AWS Lambda, Azure Functions, Google Cloud Functions, and Alibaba Cloud Function Compute), and quantitatively evaluated these several different platforms as well, AWS Lambda, Azure Functions, Google Cloud Functions, and Alibaba Cloud Function Compute), the actual performance of other serverless computing platforms were also quantitatively evaluated by means of designed benchmark platforms, i.e., micro-benchmarks and macro-benchmarks.

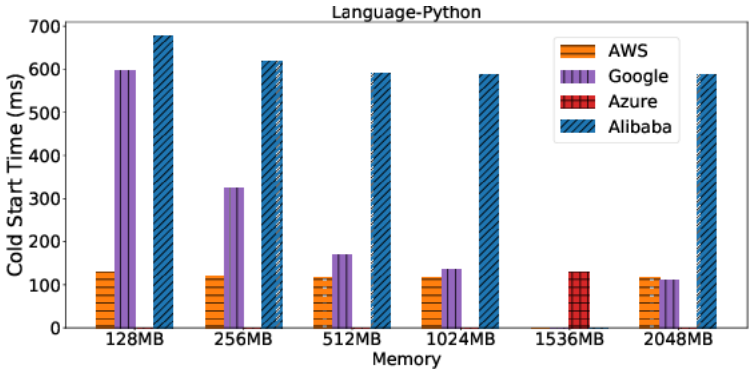
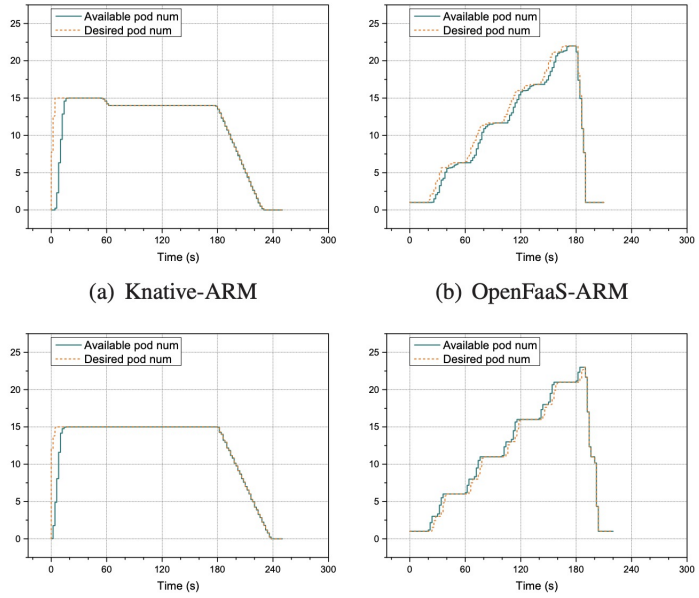


Figure 2: The distribution of the cold start time with Python on different platforms. [19]

As we can see in Figure 2, increasing memory size has varying effects on reducing cold start time across serverless computing platforms. For AWS Lamba and Alibaba Cloud Function Compute, adding more memory only increases the overhead cost.

Dong Xie, Yang Hu, et al.[20] evaluated the performance of open-source serverless computing platforms (i.e., Knative, OpenFass) on X86 and ARM architectures.



(a) Knative-ARM (b) OpenFaaS-ARM

(c) Knative-X86 (d) OpenFaaS-X86

Figure 3: Auto scaling with steady workload. [20]

We can see from Figure 3 that serverless computing on the ARM platform fails to provide a robust and stable auto scaler under a steady workload. and that Knative's auto-scaling strategy is better than

that of OpenFaas and is more adaptable to the user's needs. The authors also compare the situation under other types of workloads, which are not repeated here due to space limitations.

These studies provide valuable information about the characteristics and performance of serverless computing platforms and reveal the differences, advantages, and disadvantages between different platforms. This is an important reference for selecting and optimizing serverless computing platforms.

5 Challenges and Future Directions

In this section, we will discuss some of the challenges and future research directions for serverless computing platforms.

5.1 Challenges

The challenges that serverless computing platforms now face are diverse, such as the cold start time problem that we mentioned in our case study, and other problems faced by serverless computing platforms as mentioned by Yongkang Li and Yanying Lin et al[12]:

- Isolation: There are three ways to isolate the workloads in Linux: 1)VM; 2)Containers; 3)Language VM. Figure 4 below shows the different security methods among the three approaches, with the red symbols indicating the position of defending security. It illustrates that the container is weakly isolated because it relies on the kernel's security mechanism.

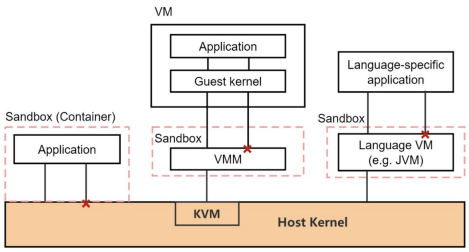


Figure 4: Different security approaches among Container, VM, and Language VM.[12]

- Scheduling Policy: The existing scheduling mechanisms designed for cloud computing are not amendable to serverless functions.
- Facility: The speed of data transfer in storage and network can greatly affect the system's performance.
- Fault Tolerance: Existing serviceless frameworks are not developed inherent strategies for fault tolerance, otherwise they would just retry when function execution fails. This can lead to errors when applications running in parallel could.

In general, challenges facing serverless computing platforms include startup latency issues, isolation issues, inappropriate scheduling policies, facility-related performance issues, and the need for improved fault-tolerance policies. These challenges require further research and development to improve the capability and effectiveness of serverless platforms.

5.2 Future Directions

Paul Castroe and Vatche Ishakian et al.[8] have proposed several future research directions.

- System-level research opportunities: An important feature capability of serverless is the ability to scale to zero, but this can lead to problems with cold starts. We can consider whether containers are the right abstractions for running serverless applications.
- Legacy code in serverless: Developers have invested a lot of effort and time into existing code, and we can consider the extent to which this legacy code can be automatically or semi-automatically broken down into smaller granularities for utilization.

- Stateful serverless: Nowadays, most serverless platforms are stateless, and we can consider whether stateful serverless platforms will emerge in the future and achieve different levels of quality of service without sacrificing scalability and fault-tolerance.
- Service level agreements (SLA): Serverless computing is poised to make it easier to develop services, but providing quality assurance of services is still difficult.
- Serverless at the edge: There is a natural connection between serverless capabilities and edge computing, as events are often generated at the edge with the increased adoption of IoT and other mobile devices. This may lead to specific requirements for redefining costs. Renchao Xie and Qinqin Tang et al.[21] mention that serverless edge computing nodes are not always physically secure as they are exposed to direct attacks through kernel vulnerabilities. The security issue here is also a potential research question.
- New serverless applications: The serverless programming model is inherently different, and we can use serverless to come up with some new solutions and applications.

Here, we discuss separately the direction of stateful serverless development, nowadays most of the serverless computing platforms are stateless. This design makes it easier to achieve elastic scaling, fault tolerance, and parallelism. However, this design does not retain any state information between function executions.

The concept of stateful serverless refers to the preservation and management of state information in a serverless computing environment. The introduction of stateful can lead to richer application scenarios: especially those that require consistent state between multiple function executions, and higher performance: stateful computing platforms can store state information in memory to access and update state data more quickly, thus improving performance. But stateful computing platforms also face some challenges, such as managing and maintaining state information that may increase the complexity of the system. Therefore, the decision of which design to use should be based on specific application requirements, and the challenges faced by each design should be carefully considered, weighing the advantages between them.

In summary, the future research directions for serverless computing include system-level investigations, handling legacy code, exploring stateful serverless platforms, improving service-level agreements, considering serverless at the edge, and exploring new applications that can benefit from the serverless programming model. These areas offer opportunities for further advancements and enhancements in the field of serverless computing. However, since this paper was published in 2019 and serverless computing platforms are rapidly evolving, the future research directions mentioned in this paper may have some limitations.

6 Conclusion

In this article, we provide an overview of the performance aspects of serverless computing. We highlight the significance of studying serverless computing performance and its benefits in terms of agility, scalability, and cost-efficiency.

In the section on performance evaluation, we discuss benchmarking as a method to evaluate the performance of serverless computing platforms. We examine a benchmarking test suite proposed by Horácio Martins and Filipe Araújo and discuss various performance evaluation tests, such as latency, throughput, scalability, and programming language impact. We also explore a benchmarking tool developed by the researchers to automate the performance evaluation process.

Next, we delve into performance modeling, focusing on the work of Nima Mahmoudi and Hamzeh Khazaei. We discuss the importance of understanding function instances and their states, such as initializing, running, and idle, as well as the concepts of cold start and warm start. We highlight the significance of performance modeling in developing accurate performance models for serverless computing platforms.

In the section on performance optimization, we present several methods to optimize performance in serverless computing. We discuss the implementation of a retry scheme, minimizing the depth of the lambda function call stack, using circuit breakers, and employing a warm-up strategy. These methods aim to improve resource availability, reduce latency, and mitigate errors in serverless applications.

In the case studies section, we examine studies conducted by Jinfeng Wen, Yi Liu, et al., and Dong Xie, Yang Hu et al. These studies provide insights into the characteristics and performance of mainstream commodity serverless computing platforms as well as open-source serverless computing platforms.

Finally, we discuss the challenges and future directions in serverless computing. We highlight challenges such as cold start time, isolation, scheduling policy, facility-related performance issues, and fault tolerance. We also present future research directions, including system-level research opportunities, legacy code in serverless, stateful serverless, service-level agreements, serverless at the edge, and new serverless applications.

Overall, this article offers a comprehensive overview of the performance aspects of serverless computing, including evaluation, modeling, optimization, case studies, challenges, and future directions. It serves as a valuable resource for understanding and improving the performance of serverless applications.

References

[1] Apache jmeter. Online. Accessed May 13th, 2020.

[2] The c4 model for software architecture. Online. Accessed May 13th, 2020.

[3] Operating Lambda: Performance optimization – Part 1 | Amazon Web Services — aws.amazon.com. https://aws.amazon.com/blogs/compute/operating-lambda-performance-optimization-part-1/. [Accessed 05-Jun-2023].

[4] Serverless Architectures with AWS Lambda: Overview and Best Practices | Amazon Web Services — aws.amazon.com. https://aws.amazon.com/blogs/architecture/serverless-architectures-with-aws-lambda-overview-and-best-practices/. [Accessed 05-Jun-2023].

[5] Serverless, the way cloud should be. Online. Accessed May 13th, 2020.

[6] David Bermbach, Ahmet Selçuk Karakaya, and Stefan Buchholz. Using application knowledge to reduce cold starts in faas services. In *Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing*, 2020.

[7] D. Bortolini and R.R. Obelheiro. Investigating performance and cost in function-as-a-service platforms. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 174–185. Springer, 2019.

[8] Paul C. Castro, Vatche Isahagian, Vinod Muthusamy, and Aleksander Slominski. The server is dead, long live the server: Rise of serverless computing, overview of current state and future trends in research and industry. *ArXiv*, abs/1906.02888, 2019.

[9] K. Figiela, A. Gajek, A. Zima, B. Obrok, and M. Malawski. Performance evaluation of heterogeneous cloud functions. *Concurrency and Computation: Practice and Experience*, 30(23):e4792, 2018.

[10] Phani Kishore Gadepalli, Gregor Peach, Ludmila Cherkasova, Rob Aitken, and Gabriel Parmer. Challenges and opportunities for efficient serverless computing at the edge. In *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, pages 261–2615, 2019.

[11] Mor Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[12] Yongkang Li, Yanying Lin, Yang Wang, Kejiang Ye, and Chengzhong Xu. Serverless computing: state-of-the-art, challenges and opportunities. *IEEE Transactions on Services Computing*, 16(2):1522–1539, 2022.

[13] Po-Ming Lin and Avi Glikson. Mitigating cold starts in serverless platforms: A pool-based approach. *arXiv preprint arXiv:1903.12221*, 2019.

[14] W. Lloyd, S. Ramesh, S. Chinthalapati, L. Ly, and S. Pallickara. Serverless computing: An investigation of factors influencing microservice performance. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pages 159–169. IEEE, 2018.

[15] Jörg Manner, Manuel Endreß, Tobias Heckel, and Gregor Wirtz. Cold start influencing factors in function as a service. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 181–188. IEEE, 2018.

[16] H. Martins, F. Araujo, and P.R. da Cunha. Benchmarking serverless computing platforms. *Journal of Grid Computing*, 18:691–709, 2020.

[17] Johann Schleier-Smith, Vikram Sreekanti, Anurag Khandelwal, Joao Carreira, Neeraja J. Yadwadkar, Raluca Ada Popa, Joseph E. Gonzalez, Ion Stoica, and David A. Patterson. What serverless computing is and should become: The next phase of cloud computing. *Commun. ACM*, 64(5):76–84, apr 2021.

[18] L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift. Peeking behind the curtains of serverless platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 133–146, 2018.

[19] Jinfeng Wen, Yi Liu, Zhenpeng Chen, Junkai Chen, and Yun Ma. Characterizing commodity serverless computing platforms. *Journal of Software: Evolution and Process*, page e2394, 2021.

[20] Dong Xie, Yang Hu, and Li Qin. An evaluation of serverless computing on x86 and arm platforms: Performance and design implications. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 313–321. IEEE, 2021.

[21] Renchao Xie, Qinqin Tang, Shi Qiao, Han Zhu, F Richard Yu, and Tao Huang. When serverless computing meets edge computing: Architecture, challenges, and open issues. *IEEE Wireless Communications*, 28(5):126–133, 2021.

Distributed Graph Neural Network Training

Yi Rong
Vrije Universiteit
y.rong@student.vu.nl

Hongyu Wu
Vrije Universiteit
wu2@student.vu.nl

Jiahui Xiong
Vrije Universiteit
xiong2@student.vu.nl

Abstract

This paper presents a comprehensive review of strategies and techniques aimed at enhancing the efficiency of distributed training for Graph Neural Networks (GNNs). GNNs have emerged as powerful deep learning models with wide-ranging applications. However, scaling these models to handle large graphs efficiently poses a significant challenge. In this work, we delve into the intricacies of distributed training methods, highlighting the specific challenges that arise when training GNNs in a distributed manner. We provide a detailed overview of common approaches and techniques that have been proposed to overcome these challenges. Additionally, we identify potential areas for future research and development in the domain of distributed training for graph data.

1 Introduction

GNNs have indeed gained widespread adoption across various real-world applications in critical domains, owing to their superior capabilities. Their versatility in handling graph-structured data has made them valuable tools in numerous fields. Social Networks: GNNs have been extensively employed in social network analysis, enabling tasks such as community detection, recommendation systems, influence modeling, and predicting user behavior. Recommendation Systems [23]: GNNs have proven effective in enhancing recommendation systems by incorporating graph-based information. They can leverage the relationships between users, items, and their attributes to provide personalized recommendations. Bioinformatics: GNNs are utilized in analyzing biological networks, such as protein-protein interaction networks and gene regulatory networks. They assist in tasks like protein function prediction, drug discovery, and disease classification. Knowledge Graphs [16]: GNNs are valuable in reasoning and knowledge representation tasks within knowledge graphs. They enable semantic understanding, entity linking, question-answering systems, and knowledge graph completion. Natural Language Processing [24]: GNNs are employed in tasks like sentiment analysis, text classification, named entity recognition, and document summarization by representing text data as graphs and capturing dependencies between words or sentences. Recommendation in E-commerce: GNNs are utilized to enhance product recommendation systems by considering user-item interactions, item similarities, and user similarities within a graph structure.

However, both industry and academia are eagerly anticipating advancements in accelerating GNN training due to the following reasons:

The rapid expansion of graph data has led to a significant increase in the time required for training Graph Neural Networks (GNNs). The proliferation of information on the Internet contributes to the continuous generation and modification of new graph data, including the establishment and dissolution of interpersonal relationships in social communication and shifts in people's preferences for online shopping. These graph structures now consist of vertices and edges reaching or even surpassing the order of billions and trillions, as reported in previous studies [20]. The growth rate of graph sizes is astonishing, exemplified by the annual growth rate of 17% in the number of vertices (i.e., users) in Facebook's social network [20]. Consequently, the training time for GNNs experiences a significant surge due to the ever-expanding scale of graph data.

2 Graph neural networks

Graph neural networks (GNNs) have emerged as the leading approach for learning over graph data [28], surpassing other algorithmic models. While deep neural networks (DNNs) have proven effective for analyzing Euclidean data like images [9], they face challenges when dealing with graph data from non-Euclidean domains due to the intricate topology and arbitrary size of graphs [4]. Furthermore, one significant drawback of deep learning paradigms, as recognized by the industry, is their limited capability for causal reasoning, which hinders the cognitive abilities of intelligent systems [1].

In response to these challenges, GNNs have become the premier paradigm for graph learning, empowering intelligent systems with enhanced cognitive capabilities. Figure 1 (b) provides an illustration of GNNs. Given graph data as input, GNNs employ forward and backward propagation to update the model parameters. The trained model can then be applied to various graph-related tasks, including vertex prediction [7] (predicting properties of specific vertices), link prediction [25] (predicting the existence of edges between vertices), and graph prediction [26] (predicting properties of the entire graph), as depicted in Figure 1 (c).

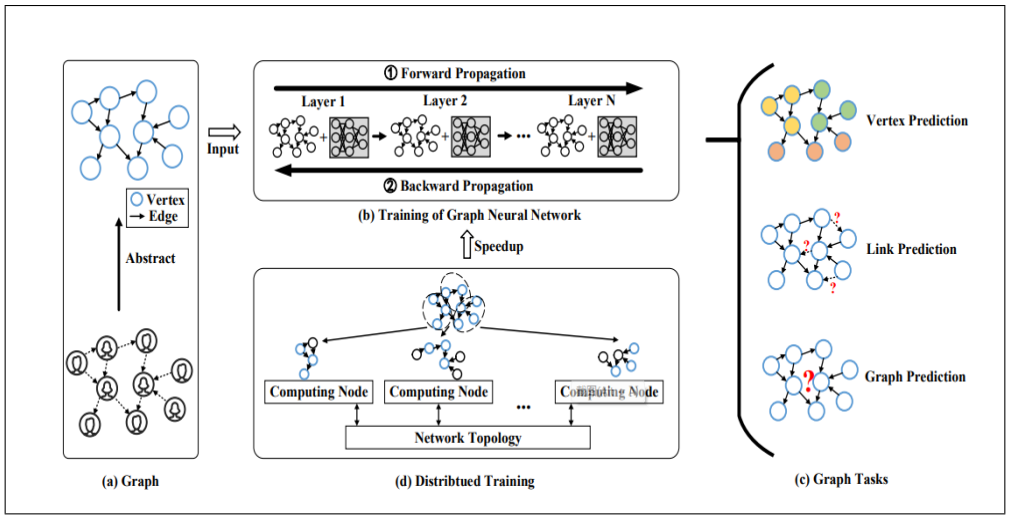


Figure 1: Distributed GNN training: (a) Illustration of graphs; (b) Illustration of training; (c) Illustration of graph tasks.[10]

2.1 Background

Graphs.A graph is a data structure composed of vertices and edges, providing a flexible framework for representing relationships among objects of varying sizes and undefined structures. This type of data, referred to as non-Euclidean data, does not possess the same highly structured form as Euclidean data, as illustrated in 2 (a) and (b). However, by utilizing vertices to represent objects and edges to denote relationships between objects, graphs offer an effective means of representing non-Euclidean data, including social networks and knowledge graphs.

Graph Neural Networks. Graph Neural Networks (GNNs) have emerged as a highly promising algorithmic model for extracting knowledge from graph data [31]. GNNs operate by taking graph data as input and learning a representation vector for each vertex in the graph. These learned representations can then be leveraged for various downstream tasks, including vertex prediction [7], link prediction [25], and graph prediction [27]. GNNs have demonstrated their efficacy in capturing complex relationships within graph data and enabling effective knowledge extraction and prediction tasks.

In the context of a graph G with an adjacency matrix A and a feature matrix X, where each row represents the initial feature vector x of a vertex v in the graph G, the lth layer of a Graph Neural Network (GNN) updates the vertex features by aggregating information from their respective neighborhoods. This process can be formalized in matrix view as follows:

$$H^l = \sigma(\tilde{A}H^{l-1}W^{l-1}) \tag{1}$$

where HL is the hidden embedding matrix and $H0 = X$, W^{l-1} is the model weights, A is normalized and σ is a non-linear function, e.g. Relu, Sigmoid, etc. The local view of GNN computation is the computation of a single vertex. Given a vertex v, the local computation of lth layer in the message passing schema can be formalized as below :

$$h_v^l = \sigma(h_v^{l-1}, \oplus_u (\varphi(h_v^{l-1}, h_u^{l-1}, h_{e_{u,v}}^{l-1}))) \tag{2}$$

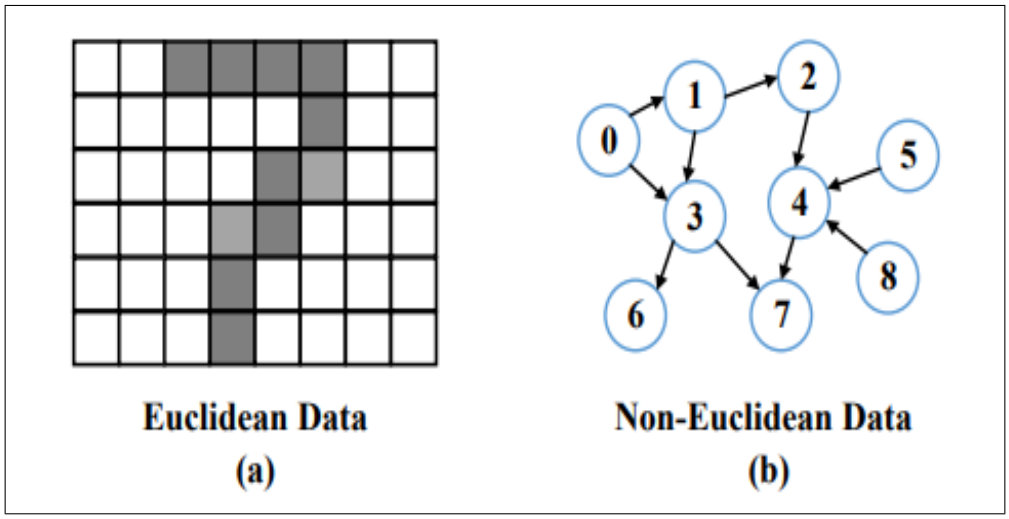


Figure 2: Illustrations of Euclidean and non-Euclidean data: (a) Image from Euclidean space; (b) Graph data from non-Euclidean space.[10]

Distributed training. Distributed training distributes the computational tasks across multiple computing devices for parallel processing, speeding up the training process and improving training effectiveness. In distributed training, there is usually a master node (or parameter server) and multiple worker nodes (or training nodes). The master node manages the model's parameters and distributes them to the worker nodes [5]. Each worker node has a portion of the training data and performs computations and optimizations using the current parameters. In each iteration, the worker nodes send the computed gradient information back to the master node, which updates the global parameters based on the collected gradients and redistributes the updated parameters to the worker nodes. This process continues iteratively until a predetermined number of training rounds or convergence criteria are reached [8].

3 Distributed GNN training

Since the limited computing resources on a single machine become the bottleneck of GNN training when handling large-scale graphs, distributed training has become a popular solution to improve the training efficiency. Proposed 1n 2019, NeuGraph[12] was the first published work of distributed GNN training. Since then, different approaches has been made to improve the efficiency of distributed GNN training in recent years.

3.1 Training Pipeline

The general training pipeline of distributed GNN training is shown in Figure 3. As illustrated in the figure, the training process can be divided into three parts: data partition, GNN model optimization, and gradient aggregation.

Data Partition. In this process, the whole graph will be divided and delivered to different workers. Comparing to traditional distributed machine learning, a major difference in distributed full-graph training is the data dependency. This is an important phase that determines the workload between workers.

GNN model optimization. This is the core phase of distributed GNN training where the training logic of the model is executed. Firstly, a computation graph will be generated by the worker according

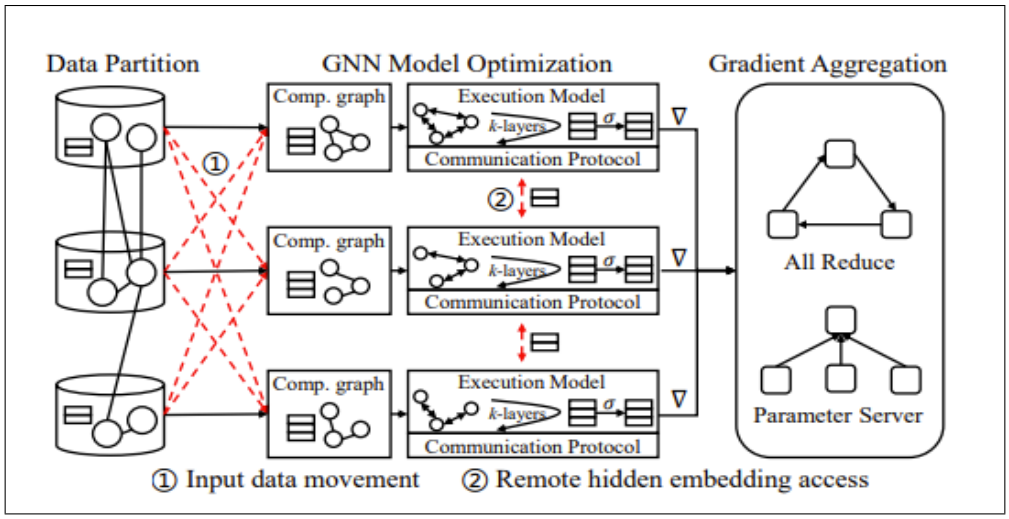


Figure 3: Training pipeline[18]

to the input pratitioned data and features. Then the GNN model will be executed on the computation graph and the communication protocol will gurantee the data exchanges with remote workers.

Gradient aggregation. In this process, all the latest gradients from different workers will be collected so that the global gradients can be obtained and used to update the model parameters. [18] pointed out that the existing gradient aggregation techniques in classical distributed machine learning can be directly used in distributed GNN training process.

As shown in Figure 4, generally, traning methods of GNNs can be classified into full-graph training and mini-batch training, depending on whether the whole graph is involved in a traning round.

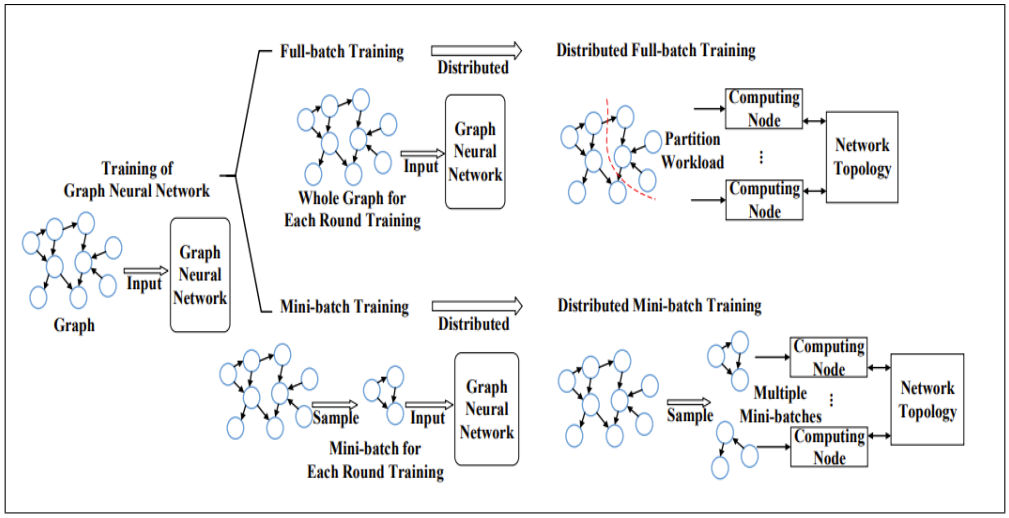


Figure 4: Training methods for GNNs and their distributed implementations.[10]

3.2 Full-graph Tranining

Full-graph training is an approach that utilizes the information of the whole graph to update the model parameters in every training round. Before the training process, the whole graph is partitioned into different sub-graphs and delivered to different workers. Due to the cross edges between sub-graphs, simply training on a sub-graph may cause a loss of accuracy. Therefore, constructing a proper data partition algorithm and a communication protocol between workers is important.

3.3 Mini-batch Training

Similar to full-graph training, mini-batch training is another distributed implementation of GNN training. The batch generation process could be further classified into partition-based mini-batch generation and sampling-based mini-batch generation.

Sampling-based mini-batch training employs sampling techniques to create mini-batches over large graphs. Most of the existing distributed GNN systems, like DistDGL[29], BGL[11], AliGraph[30] follow this idea. In such a process, a worker will frequently access the data from other workers, which may lead to massive computational expenses.

Partition-based mini-batch generation is to only train the GNN model on a local partition, which could avoid massive communication between workers. PSGD-PA[15] is a straightforward implementation of this idea. However, the efficiency improvement in the partition-based approach can lead to the loss of accuracy and the uncertainty of convergence[6]. Therefore, finding a balance between efficiency and accuracy is crucial in distributed mini-batch training.

4 Challenges

In the context of distributed training, the system will face several challenges that may lead to inefficacy during the training process. In this section, we categorize these challenges into three distinct perspectives, each of which offers a different aspect of the distributed training paradigm. By dissecting the challenges from multiple perspectives, we gain a comprehensive understanding of the current domain and develop our research question.

4.1 Challenge 1: Computation approach

As shown in Figure 3, in the stage of model optimization, two sub-stages are involved: computation graph generation and computation graph execution. The execution model, which is the key computation strategy, arranges the operation within these sub-stages to get optimal efficiency. However, In the case of distributed mini-batch training, the sampling and feature extraction operations play a pivotal role in dictating the overall training efficiency, leading the computation graph generation to be resource-intensive. Conversely, in the context of distributed full-graph training, the execution of the computation graph becomes intricate due to the presence of data dependencies among the various workers involved.

4.2 Challenge 2: Batch generation

The generation of batches plays a pivotal role in the training of deep learning models, as highlighted by Masters et al. in their study on the subject[14]. In the domain of distributed training for deep learning, the utilization of mini-batch generation has been shown to accelerate the training process, as demonstrated by Serizawa et al.[17]. However, when this concept is extended to distributed GNNS , the generation of batches poses unique challenges owing to the intricate nature of the underlying data structure. To be more specific, the workload assigned to each node is imbalanced, leading to a notable impact on both the training efficiency and the ultimate accuracy of the GNN model.

4.3 Challenge 3: Ineffective Data Partitioning

In the distributed setting, prior to inputting data into the model, it is important to store the data across the hardware infrastructure. However, this storage process often encounters challenges associated with workload imbalances among the individual nodes, arising from the preparation strategy employed.

When preparing the data for storage in a distributed environment, certain factors can lead to workload imbalances across the nodes. The choice of strategies, such as random partitioning or graph-based partitioning, can influence the distribution of data among the nodes. Depending on the nature of the data and the selected strategy, it is possible that certain nodes may be burdened with a significantly higher volume of data compared to others. Consequently, this imbalance in the workload distribution can introduce inefficiencies and hinder the overall performance of the distributed system.

5 Improve training efficiency approaches

In the preceding section, we expounded upon the challenges inherent in the distributed training of Graph Neural Networks (GNNs).To investigate the possible approach to overcome these challenges. In this section, we put forth a research question: What are the viable strategies and techniques that can be harnessed to enhance the efficiency of the distributed training process for GNNs deployed in distributed environments? We will answer them from three different perspectives that correspond to the challenges we discussed before with real word application.

5.1 Computation approach

In distributed settings, the execution of computation graphs is carried out by a group of workers. When performing distributed mini-batch GNN training, parallelizing the execution of computation graphs for each mini-batch is straightforward, as each graph is fully contained within a single worker. However, when dealing with distributed full-graph or large-batch GNN training, the execution of a computation graph needs to be partitioned among multiple workers. This presents a challenge due to the data dependency within the computation graph, making it difficult to achieve high efficiency in GNN training.

To tackle this challenge, several computation graph execution models for GNN distributed training have been proposed, which aim to optimize the execution process and improve the overall training efficiency. By partitioning the computation graph across workers and managing the data dependencies, these models minimize communication overhead and maximize parallelization. We categorize them by means of implementation, namely chunked-based, and one-shot.

5.1.1 Chunked-based execution

Initially, each machine receives the computation graph for layer 1 and performs the forward pass in a model parallel fashion, computing partial activations based on the partitioned input features. The partial activations are then aggregated through a reduce operation, and the computation switches to data parallelism.To obtain the final activations, the forward pass concludes with data-parallel computation of the embedding. The backward pass involves global gradient synchronizations , after which the error gradient is pushed to all machines, and the computation switches back to model parallelism for local backward pass computation.

The process of partial aggregation occurs sequentially, with each partial result being accumulated towards the final aggregation result in a step-by-step manner. In the NeuGraph framework [13], a 2D partitioning method is employed to generate multiple edge chunks, resulting in the partitioning of vertex neighborhoods into sub-neighborhoods accordingly. Each worker is assigned the task of aggregating specific vertices and sequentially processes the corresponding edge chunks to compute the final aggregation result.

5.1.2 One-shot execution

The one-shot execution model is widely adopted in GNN training, where each worker pre-fetches the necessary neighbor features for local graph aggregation and performs the aggregation in a single step. Remote information may require inter-worker communication. Within the local perspective, specific workers are assigned to target vertices, responsible for gathering neighbor features and performing aggregation. This model streamlines the process, reducing communication rounds and enabling faster computation by consolidating neighbor information in advance. By minimizing the overhead of accessing remote information, the one-shot execution model enhances the efficiency of GNN training.

Wan et al. [21] have proposed a notable distributed Graph Neural Network (GNN) framework known as BNS. In the context of distributed training, BNS employs a partition parallelism strategy, wherein the original graph is partitioned into smaller subgraphs. This partitioning enables localized training on individual accelerators/nodes, while simultaneously facilitating the exchange of dependent node features across subgraphs. Within each subgraph, inner node sets are defined, while boundary node sets encompass dependent nodes from other subgraphs. Consequently, the communication of features and gradients between subgraphs becomes essential for each Graph Convolutional Network (GCN) layer. The model is updated through weight gradient sharing among the partitions, thereby facilitating the iterative optimization process in the distributed training of GNNs.

5.2 Batch generation

5.2.1 Cashed-based generation

The distributed version of existing sampling-based GNNs involves implementing distributed sampling to generate mini-batches from large graphs. This approach is commonly adopted in various distributed GNN systems. However, in a distributed environment, the use of basic samplers on individual workers often leads to significant communication overhead due to frequent data access across workers. Furthermore, in multi-GPU settings, where graphs and features may be centralized and stored in CPU memory, each GPU still requires extensive CPU-GPU data movement to access the mini-batch data.

To address these issues, the cache strategies have been proposed. For example, DistDGL replicates remote neighbors of boundary vertices locally to eliminate communication during mini-batch generation. Other approaches, such as using importance metrics or static GPU caches, aim to strike a balance between communication reduction and storage overhead. Weighted reverse PageRank is used to identify frequently accessed vertices, and their features are cached in GPU memory, while less frequently accessed vertices are stored in CPU memory. Additionally, the use of NVLink and PCIe bandwidths is leveraged to optimize caching efficiency by replicating the hottest vertex features and scattering the next hottest data across multiple GPUs.

5.2.2 Partition-based generation

The distributed GNN training pipeline, as depicted in Figure 3, involves the initial partitioning of the graph. By adopting a training strategy where each worker exclusively handles the GNN model training for its local partition, the need for extensive communication can be mitigated. This training approach treats each partition as a mini-batch, thus referred to as a partition-based mini-batch.Each mini-batch is processed by a worker during the training phase. By partitioning the graph and assigning specific partitions to individual workers, the batch generation process becomes more efficient and scalable.

One main limitation is an unbalanced partition when dealing with a cross-with-edge situation, which can result in a loss of model accuracy. To address this, researchers have introduced subgraph expansion techniques. Subgraph expansion preserves the local structural information of boundary vertices by replicating remote vertices locally. One common approach is to use METIS algorithm[22], to obtain a collection of subgraphs. Each subgraph is then expanded by incorporating the one-hop neighbors of vertices that do not belong to the subgraph. This expansion strategy helps capture important connections and improves the overall accuracy of the model. By incorporating additional vertices from neighboring subgraphs, the expanded subgraphs provide a more comprehensive representation of the underlying graph structure, leading to enhanced accuracy in GNN training.

5.3 Data Partition

For the data partition in the pipeline of distributed training, we divided them into 3 main categories according to their techniques of implementation (shown in Figure5). We will discuss them respectively

5.3.1 Graph based partition

Graph Neural Networks (GNNs) are a form of graph computation task that can benefit from classical graph partition methods. In the context of distributed mini-batch GNN training, achieving workload balance among workers entails the equitable distribution of train vertices, corresponding to the number of batches per epoch. This introduces new optimization objectives.

Graph partition problem was proposed in the DistDGL framework by formulating it as a multi-constraint partition problem, which balances training/validation/test vertices and edges within each partition. They leverage the multi-constraint mechanism employed by METIS to achieve customized graph partition objectives for both homogeneous and heterogeneous graphs.Furthermore, they extend this mechanism to a two-level partitioning strategy that accommodates scenarios with multiple GPUs and machines while ensuring computational balance

In the context of distributed full-graph GNN training, the goal is to balance the workload of each GNN layer across a set of workers while minimizing communication for embedding. To capture the

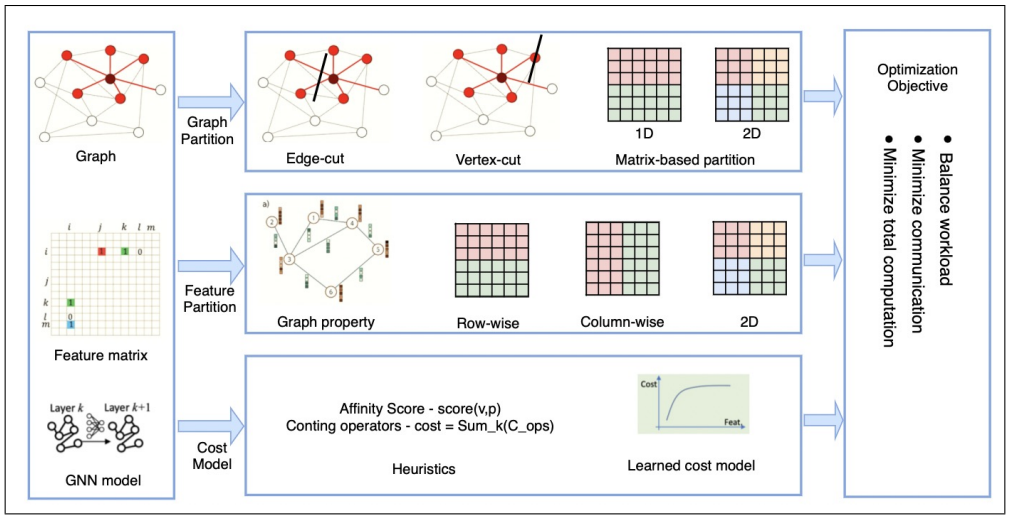


Figure 5: Data partition approaches in the distributed training of GNNs

intricate costs associated with diverse GNN models, researchers utilize learning-based cost models. These models provide insights into the computational requirements of each partition. In conjunction with these model-aware cost models, existing graph partition methods are employed.

Wang et al.[19] estimate the computation cost of a partition and employ an application-driven graph partition method to generate a workload balancing plan. This approach adapts dynamically to mitigate workload imbalances and minimize communication costs. Besides, in terms of vertex-cut, leverage 2D Cartesian vertex-cut techniques to enhance scalability.

5.3.2 Feature based partition

In the context of Graph Neural Networks (GNNs), partitioning the features is crucial as they represent important data. Many distributed GNN training solutions consider features as vertex properties within the graph and partition them along with the graph structure. For instance, if the graph is partitioned using the edge-cut method, each vertex's feature is stored in the corresponding partition where the vertex resides, resulting in a row-wise partitioning of the feature matrix.

However, due to the distinct processing patterns of features compared to the graph structure, several approaches have emerged that partition the feature matrix independently from the graph. Tripathy et al. [22] adopt the 2D partition method and row-wise method, respectively, to partition the feature matrix. They align the feature matrix partitioning with the graph partitioning, while Vasimuddin et al. ensure that each vertex has access to the complete feature locally by using replication.

5.3.3 Cost model based partition

In addition to the topology of the graph, partitioning techniques can also be based on modeling the communication overhead and workload associated with each node in the graph. These approaches can be broadly categorized into two main categories: heuristic-based and learning-based methods.

heuristic-based methods

Many Graph processing frameworks apply heuristic-based methods and strategies to achieve workload balance in the data partition, such as GraphX [2] and Gemini [32]. These frameworks statically partition input graphs by optimizing heuristic objective functions, namely affinity scores, The affinity scores play a crucial role in determining the most suitable partition for each vertex or block, ensuring an optimized partitioning outcome. It typically considers factors like the number of edges spanning different partitions While these objective functions yield favorable outcomes for data-intensive graph processing, they fall short in the case of compute-intensive graph neural networks (GNNs) due to the substantial variations in per-vertex computation loads.

learning-based methods Applying machine learning techniques on cost models in GNNs data partition has gained attention. By incorporating both static graph structural information and runtime statistics of GNN workloads, the learning-based model offers a more precise estimation of the cost compared to heuristic models. In the case of ROC[6], an online linear regression-based algorithm is employed to achieve balanced partitioning for GNN training and inference. This is achieved by jointly learning a cost model capable of predicting the execution time of the GNN model on diverse graphs.

6 Feature work

Distributed Graph Neural Networks (GNNs) provide a general solution for training GNNs at scale. In addition to the techniques and systems mentioned earlier, there are several other fascinating and emerging research areas in distributed GNN training. We discuss some interesting directions in the following

Scalable Graph Sampling. Graph sampling techniques aim to address the scalability challenges posed by extremely large graphs. By selecting a representative subset of nodes or edges, distributed GNN training can be performed on a reduced graph, significantly reducing computational and memory requirements. Research focuses on developing efficient and effective graph sampling methods for distributed GNN training.

Dynamic and Heterogeneous Graphs. Real-world graphs are often dynamic and heterogeneous, with evolving structures and diverse node and edge attributes. While dynamic graph neural networks (GNNs) have garnered considerable interest in research, it is noteworthy that there is currently a lack of literature dedicated to the specific topic of distributed dynamic GNN training. The inherent dynamism encompassing both features and structure introduces fresh challenges to conventional solutions within a distributed environment. Graph partitioning techniques need to swiftly adapt to the fluctuating vertices and edges, while still adhering to load balancing and communication reduction requirements [3]. Furthermore, the modifications made to the graph structure lead to a decrease in the cache hit ratio, which greatly impacts the overall performance of end-to-end GNN training.

7 conclusion

In conclusion, we provide a comprehensive investigation of the distributed training pipeline for Graph Neural Networks (GNNs). We have extensively discussed the fundamental concepts of GNNs and their application in distributed training. Additionally, we have identified and examined the challenges associated with data partitioning, batch generation, and computation approaches within the distributed training pipeline. Various approaches and techniques to enhance training efficiency in response to these challenges have been reviewed. Furthermore, we propose promising areas of further research, such as Scalable Graph Sampling and Dynamic and Heterogeneous techniques, which warrant further investigation to address the identified challenges and advance the field of distributed training for GNNs.

References

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[2] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica. Graphx: Graph processing in a distributed dataflow framework. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 599–613, 2014.

[3] M. Guan, A. P. Iyer, and T. Kim. Dynagraph: Dynamic graph neural networks at scale. In *Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, pages 1–10, 2022.

[4] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. v. Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 1024–1034, 2017.

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia (ACM MM)*, 2014.

[6] Z. Jia, S. Lin, M. Gao, M. Zaharia, and A. Aiken. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems*, 2:187–198, 2020.

[7] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, 2017.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, 2012.

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[10] H. Lin, M. Yan, X. Ye, D. Fan, S. Pan, W. Chen, and Y. Xie. A comprehensive survey on distributed training of graph neural networks, 2022.

[11] T. Liu, Y. Chen, D. Li, C. Wu, Y. Zhu, J. He, Y. Peng, H. Chen, H. Chen, and C. Guo. {BGL}:{GPU-Efficient}{GNN} training by optimizing graph data {I/O} and preprocessing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 103–118, 2023.

[12] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou, and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 443–458, Renton, WA, July 2019. USENIX Association.

[13] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou, and Y. Dai. Neugraph: Parallel deep neural network computation on large graphs. In *2019 USENIX Annual Technical Conference*, pages 443–458. USENIX Association, 2019.

[14] D. Masters and C. Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[15] M. Ramezani, W. Cong, M. Mahdavi, M. T. Kandemir, and A. Sivasubramaniam. Learn locally, correct globally: A distributed algorithm for training graph neural networks. *arXiv preprint arXiv:2111.08202*, 2021.

[16] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[17] K. Serizawa and O. Tatebe. Accelerating machine learning i/o by overlapping data staging and mini-batch generations. In *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, pages 31–34, 2019.

[18] Y. Shao, H. Li, X. Gu, H. Yin, Y. Li, X. Miao, W. Zhang, B. Cui, and L. Chen. Distributed graph neural network training: A survey, 2022.

[19] Z. Song, Y. Gu, J. Qi, Z. Wang, and G. Yu. Ec-graph: A distributed graph neural network system with error-compensated compression. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 648–660. IEEE, 2022.

[20] A. Tripathy, K. A. Yelick, and A. Buluc. Reducing communication in graph neural network training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 70. IEEE/ACM, 2020.

[21] C. Wan, Y. Li, A. Li, N. S. Kim, and Y. Lin. Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. *Proceedings of Machine Learning and Systems*, 4:673–693, 2022.

[22] Z. Xue, Y. Yang, M. Yang, and R. Marculescu. Sugar: Efficient subgraph-level training via resource-aware graph partitioning. *arXiv preprint arXiv:2202.00075*, 2022.

[23] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Y. Guo and F. Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, KDD 2018*, pages 974–983. ACM, 2018.

[24] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

[25] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pages 5171–5181, 2018.

[26] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 4438–4445. AAAI Press, 2018.

[27] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 4438–4445. AAAI Press, 2018.

[28] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2022.

[29] D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang, and G. Karypis. Distdgl: distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*, pages 36–44. IEEE, 2020.

[30] R. Zhu, K. Zhao, H. Yang, W. Lin, C. Zhou, B. Ai, Y. Li, and J. Zhou. Aligraph: A comprehensive graph neural network platform. *arXiv preprint arXiv:1902.08730*, 2019.

[31] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang. Relation structure-aware heterogeneous graph neural network. In *2019 IEEE International Conference on Data Mining*, pages 1534–1539. IEEE, 2019.

[32] X. Zhu, W. Chen, W. Zheng, and X. Ma. Gemini: A computation-centric distributed graph processing system. In *OSDI*, volume 16, pages 301–316, 2016.

Distributed Cloud-based ML/DL Workflows

Shreyas Patil
Universiteit van Amsterdam
Amsterdam, Netherlands
shreyas.patil@student.uva.nl

Shreya Ghose
Universiteit van Amsterdam
Amsterdam, Netherlands
shreya.ghose@student.uva.nl

Madhur Pawar
Universiteit van Amsterdam
Amsterdam, Netherlands
madhur.pawar@student.uva.nl

Abstract

Resource management in machine learning pipelines can be difficult, and frequently results in under or over-provisioning. The resource needs for the pipeline's various phases vary, and the absence of DevOps specialists might make it difficult to deploy software quickly and share models. This review paper explores various MLOps frameworks and assesses them based on factors including features, architecture, performance, and integration potential. The characteristics, benefits, and drawbacks of the frameworks and respective studies conducted on frameworks like CIRRUS, DEEP, CloudFlows, Distributed GraphLab, and Kubeflow are thoroughly discussed.

1 Introduction

The paradigm of serverless computing has been increasingly used in the industry primarily due to reduced effort of operations and management of resources. The research topic of usage of this paradigm for developing and deploying machine learning-based software systems is still in the early stages as there are challenges involved in its adoption. Such systems have huge memory and compute requirements, while serverless architectures are built to support smaller workloads on each instance that can be scaled. The challenge of using the serverless with distributed data also exists as each serverless instance has limited memory and storage, hence the coordination of distributed computing in each instance and the storage has to be well established (1).

Despite the challenges involved with the industry moving to cloud systems, it could be advantageous to data scientists. Since they have limited experience and knowledge about the provisioning and management of resources, it could benefit them primarily if the management of resources is handled by the workflow frameworks and they can focus on the data science part. This has led to the rise of a new area of specialized role MLOPS whose role is to act as an interface between data scientists, data engineers, and infrastructure management, with a focus on adopting the DevOps approach to the development of machine learning resource-intensive projects (2).

In the paper, we aim to review the existing frameworks for managing workflows of machine learning and deep learning systems on the cloud. With the increasing popularity and need for DevOps-oriented execution of data science projects, it is key to know the different criteria based on which different available frameworks can be evaluated and chosen for the projects. In the paper, we also highlight the key differences between them and provide the cost, maintainability, and scalability aspects of frameworks. We also try to answer the question 'How do different cloud-based workflow frameworks solve the challenges in Machine Learning/Deep Learning (ML/DL)?'

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Challenges and Workflow in ML systems

ML pipelines are iterative in nature and have a variety of stages which make the end-to-end workflow complex in terms of resource provisioning and management. Usage of VMs for developing ML systems has been done previously but it creates 2 challenges: first, it can lead to over-provisioning, and second, it adds the responsibility of managing and configuring resources to users for every project. The existing ML frameworks such as Spark (3) are capable of handling distributed computing and storage which process all data in memory, but it does not leverage the serverless paradigm as such managing the resources is not easy and often the problem of under-provisioning or over-provisioning still persists. As the resource requirements are different in different stages of the ML pipeline which typically includes data pre-processing, model training, and model tuning the chances of over-provisioning and under-provisioning exist.

Generally, the training part requires high compute resources along with the training data being distributed while the validation and prediction phases require less. Furthermore, some algorithms require all data upfront while others process it in batches. The process of deploying and model sharing also requires DevOps knowledge which often the data scientists do not have and can reduce the speed with which the predictions can be made if there is no specialized person in that area to rely on (4).

A typical pipeline is as seen in figure 1 where there are multiple stages and the complexity of such workflows is not only due to resource management but also involves managing the different artifacts in addition to the ML model. The task of data versioning, data quality checks, and model versioning makes it even more complex to manage the entire pipeline and to keep it updated with the latest data and predictions which becomes particularly challenging when multiple teams are involved in different parts of the pipeline.(5)

With a wide range of products and tools available for use for different tasks of the workflow, it becomes necessary to evaluate the different criteria such as expertise of the team in the programming language, cost, integration capabilities with the existing codebase or tools/tech (6). The paper (7) emphasizes the need for continuous integration and continuous delivery to maintain and update the different artifacts of the pipeline in an iterative manner just as it is done in traditional software projects. The paper (8) highlights that the ML component is only a small part of the entire project and the way the other components are managed decides on the viability and success of a ML project.

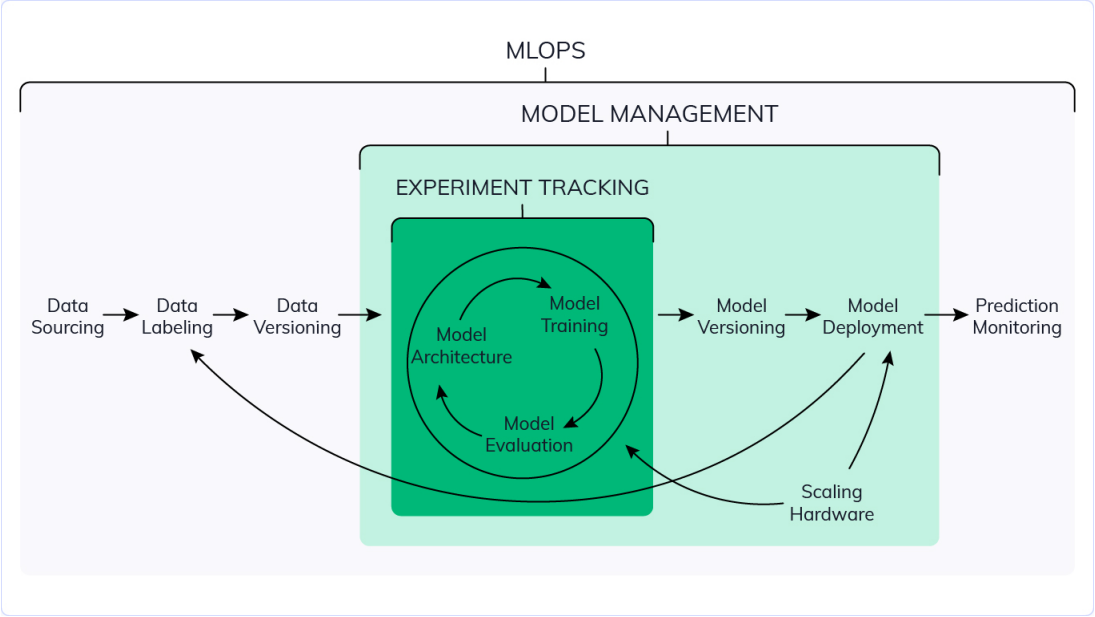


Figure 1: Typical MLOps Workflow

3 Review of MLOps Tools

3.1 CIRRUS

CIRRUS is a distributed ML training framework that is based on and leverages the serverless infrastructure offered by AWS Lambda. The paper on CIRRUS (9) mentions that there is a limit on the resources (CPU, 3GB memory, 512MB storage) that is available with Lambdas. These constraints prevent the usage of existing ML frameworks such as Tensorflow and Spark on serverless infrastructure directly. Additionally, lambda functions are not designed to share data among the executing instances which demand the use of a mechanism to have low latency and high throughput storage which is optimized for ML workloads. To overcome the limitation of existing lambda constraints it is designed as a wrapper around the lambdas and makes use of a parameter server model which combines the benefits of working with a serverless environment and ML processing frameworks. Thereby providing the convenience of resource management as well as the necessary resources to handle ML.

Implementation CIRRUS is implemented by the following four components that communicate through TCP connections:

Front-end: The user can interact with the system and visualize the progress of the workloads with the Python front-end component.

Data store: A distributed data store is used to share data between different workers executing inside lambdas. It is deployed in cloud VMs to achieve low latency and maximize system updates/sec for model updates during training. It leverages AWS S3 for storing data. It initially serializes the data and partitions it into similar-sized partitions which further speeds up the execution during training and hyperparameter tuning stages which reduces the computation load required to be executed inside lambdas.

Client back-end: The client back-end performs a number of tasks such as parsing training data, launching Cirrus workers, managing the distributed data store, and returning results to the Python front-end once computations are complete.

Runtime: It provides a lightweight runtime environment that can be executed inside lambdas which can execute ML algorithms. To overcome the limitation of storage and memory of lambdas the frameworks stream data to lambdas in mini-batches of data. It uses a server that specifically stores the gradients computed by different workers and enables the usage of stochastic gradient descent in a distributed manner as depicted in the figure 2.

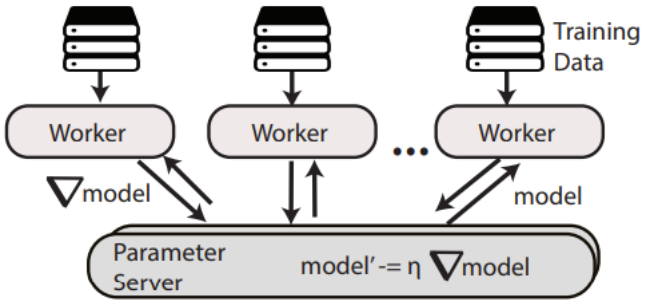


Figure 2: Distributed Stochastic Gradient Descent in CIRRUS

Features CIRRUS provides an API that helps developers construct a pipeline for preprocessing, feature engineering, and parameter tuning at scale. It assists in the process of hyperparameter search since this is computationally expensive and provides a visual dashboard that allows the user to monitor the loss function and terminate the experiment individually if the result is not going to be optimal, thus saving additional resources from being wasted.

Strengths CIRRUS is highly scalable across the dimensions of storage, computing, and shared memory. Scalability across storage is achieved by splitting data in S3 into medium-sized objects. Linear compute scalability can be achieved by streaming input data and computing gradients in parallel up to 600 workers. (9) The framework's implementation is also highly abstract: the frontend abstracts all the details from developers and the backend abstracts the management of lambdas from the frontend algorithms and keeps a list of active lambdas and connections to the AWS Lambda API.

Weaknesses The framework was tested and compared with others such as PyWren (10) and Spark (3) but the execution of CIRRUS was done on VMs that have different configurations than that of PyWren and Spark. This provides an unclear comparison of efficiency. The study (9) does not provide real-world case studies or practical examples of CIRRUS' deployment in actual ML workflows. Including such case, studies would demonstrate the applicability and effectiveness of the proposed framework in real-world scenarios. Moreover, the study does not thoroughly discuss the limitations of CIRRUS or the potential trade-offs associated with its design choices.

3.2 DEEP

'Designing and Enabling E-infrastructures for intensive Processing in Hybrid Data Clouds' also referred to as DEEP, has implemented a framework that enables training and sharing of machine learning models over a distributed infrastructure. The paper also analyses previously existing frameworks in this area and argues that most of the frameworks do not provide end-to-end workflow management capabilities including deployment and leveraging the cloud-based services. While providing the features to manage different phases of the ML pipeline it also provides flexibility to the users to use a library or tool specific to their requirements. (4)

The user can start by using an existing model or develop their own. The model is then containerized using a Docker container with all the dependencies and deployed inside the infrastructure. DEEP provides API which enables it to work with a serverless framework such OpenWhisk. The API provides an interface that enables the training, monitoring, testing, and evaluation of containerized applications. The framework supports setup on a wide range of platforms such as HPC systems, orchestration engines Kubernetes and serverless frameworks such as OpenWhisk. This enables high portability of the model on a wide range of infrastructures including the cloud.

Architecture DEEP is based on serverless architecture as can be seen in figure 3 wherein the model is encapsulated as a function. The components are as follows:

DEEP Open Catalogue: Users and user communities can browse, share, store, and download ready-to-use machine learning and deep learning modules in this marketplace. It also offers extra components like complicated application topologies and model and application-related metadata.

DEEP Learning Facility: This module coordinates and orchestrates overall model training, testing, and assessment, selecting appropriate Cloud resources based on computational and storage resources.

DEEP as a Service: DEEPaaS allows users to deploy and serve an already trained model from the catalog as a service, allowing third-party users to access the model's capabilities and knowledge.

Storage and data services: All data assets are stored in this module.

Features The framework puts emphasis on sharing the model since that is an effective method to disburse knowledge in the scientific community and enable collaboration, which other frameworks do not seem to pay much attention to. Moreover sharing a model also enables reproducibility of experiments which is desirable in the scientific community. DEEP's UI layer, the DEEP-HPC Portal, offers a user-friendly web-based interface that lets users interact with the infrastructure, submit processing tasks, track task status, and view outcomes. The interface makes the infrastructure more accessible to researchers with varied levels of technical skill.

Strengths DEEP tries to reduce the dependency of machine learning pipelines on users' resources. The usage of Docker containers allows for the delivery of customized environments adapted to the demands of the user, as well as maximum portability of the runtime environments. DEEPaaS API can also deploy modules on any infrastructure. Continuous Deployment implementation is done on DEEPaaS to quickly integrate newly launched or updated models. DEEP encourages user and

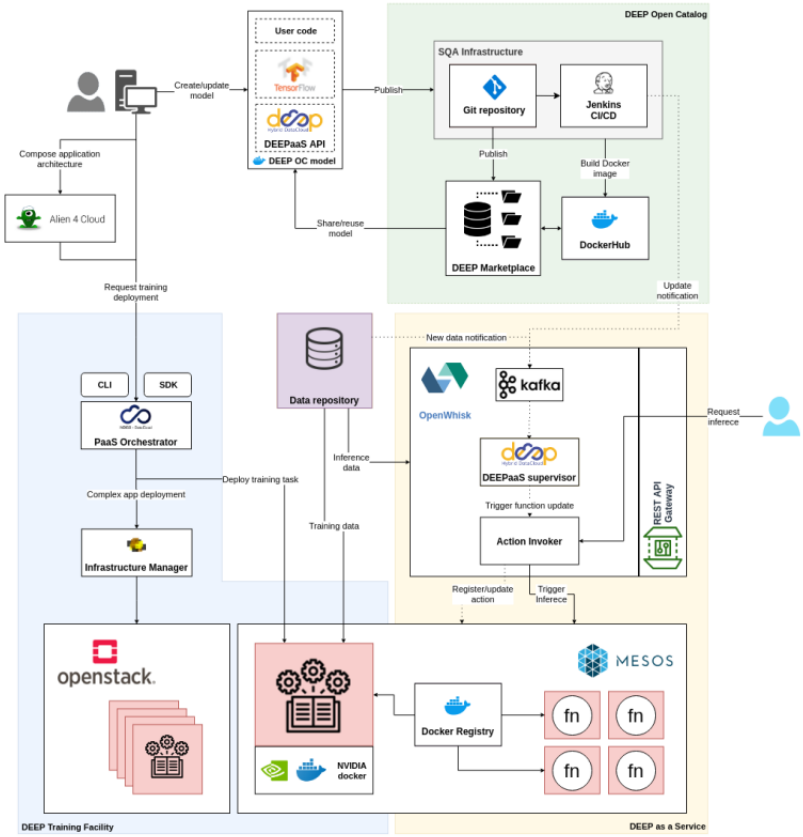


Figure 3: DEEP architecture

researcher collaboration and sharing. It enables users to collaborate on common scientific concerns by providing means for sharing workflows, data, and processing outcomes. This promotes information exchange and repeatability, as well as collaborative research initiatives. The study also makes use of an excellent case study on image-based classification to demonstrate the usage of the platform.

Weaknesses DEEP does not yet support distributed training, hence contributing to time and resource overheads. Distributed training would help improve the efficiency of the hosted models. The study also does not provide any empirical evidence of DEEP's performance as compared to other platforms.

3.3 Kubeflow

Kubeflow is a cloud-based machine learning platform built on Kubernetes, an open-source container orchestration system. It provides a range of tools and components to streamline the development, deployment, and management of machine-learning models in a scalable and portable manner.

Architecture The Kubeflow Architecture serves as the foundation for the Kubeflow platform, a toolkit designed for the development and deployment of machine learning (ML) workflows on Kubernetes. The architecture comprises several key components and elements.(11)

Python SDK: The Kubernetes domain-specific language (DSL) can be used by users to create components or define pipelines, which is why the Python SDK is essential to the Kubeflow Architecture. Users can specify the logic and structure of their ML workflows by utilizing the Python SDK.(12)

DSL Compiler: The Python code used in the pipelines must be translated by the DSL compiler into a static configuration format, which is represented by a YAML file. The pipeline configuration is stored in a static, easily readable manner using this conversion procedure.(12)

Kubeflow Configuration Interfaces: Users can specify and install the machine learning (ML) tools necessary for their workflows using Kubeflow's configuration interfaces. These configuration interfaces make it easier to integrate different ML frameworks and tools into the Kubeflow platform.(11)

Kubeflow Components: The Kubeflow components are the essential services that offer specific functionalities for different stages of the ML workflow. These components encompass a range of capabilities, including data preparation, model training, prediction serving, and monitoring. They form the building blocks of ML workflows within the Kubeflow ecosystem.(13)

Features Key features of Kubeflow include pipelines for defining and executing end-to-end machine learning workflows, capabilities for distributed training and serving of models, tools for experimentation and hyperparameter tuning, data management functionalities for versioning and preprocessing, and model versioning and deployment capabilities. Kubeflow also offers visualization and monitoring tools to track metrics and monitor resource utilization. One of the notable advantages of Kubeflow is its integration with Kubernetes, enabling users to leverage the scalability and portability of Kubernetes for running machine learning workloads across different environments.(14)

Evaluation (15) explores the deployment of machine learning models using Kubeflow on cloud platforms. The study compares baseline cloud architectures to Kubeflow's capabilities and investigates challenges faced during the deployment process. In the experiment, baseline cloud architectures are set up, and Kubeflow's capability to standardize and automate machine learning workflows is assessed. The deployment procedure is investigated on IBM Cloud and Google Cloud Platform (GCP) utilizing MiniKF and other configuration techniques. Two strategies—a code-based strategy and an end-to-end (E2E) strategy—are contrasted. The study includes an examination of hyperparameter tuning using Katib, model serving through a Flask-based application, and the utilization of Kubeflow add-ons such as Istio and KServe. Experiments on NYU Greene Cluster, Kubernetes on IBM Cloud, Kubeflow on IBM Cloud, and Kubeflow on Google Cloud Platform were performed. The results showed that the approaches using Kubeflow performed better in terms of inference time, indicating that Kubeflow is a good choice for model serving.

The authors also contrasted how well Kubeflow performed on Google Cloud Platform and IBM Cloud. Katib and Model Serving were used to analyze the runtimes of both customized models and an entire pipeline. The results indicated that Kubeflow on Google Cloud performed slightly faster on average.

It was noted that the IBM Cloud's Kubeflow-powered inference model had the quickest inference times, potentially as a result of the dedicated VPC and improved network speed. It was also observed that the total time required to run the pipeline was less on GCP, which might be related to the more potent cluster and less intense competition for resources.

Using Kubeflow and setting up a cluster were simpler on GCP, mostly because of better documentation and more capabilities. There are several difficulties with Kubeflow, including the setting of the initial installation and authentication, the complexity of upgrading components, and obsolete documentation with broken links that prevent the framework from being widely used.

Strengths The experiment examines a number of Kubeflow-related factors, such as setup, deployment strategies, performance, and the tool's capabilities and restrictions. It offers comprehensive measurements and comparisons to assess Kubeflow's performance.

Insights are derived from the results and recommendations are provided based on the findings. It discusses the inference time, performance across different clouds, ease of use, and challenges faced during the experiment. These insights can guide future researchers and users in making informed decisions.

Weaknesses While the experiment acknowledges Kubeflow's limits, it does not go into great detail on any potential problems or difficulties that may have arisen during the trial itself. Providing a more comprehensive discussion of limitations would add depth to the analysis.

Although the experiment briefly covers Kubernetes and Kubeflow's scalability characteristics, it does not offer a thorough review of the tool's scalability in real-world situations. Scalability is essential for properly deploying machine learning models, especially given the growing volume of data and internet traffic.

3.4 ClowdFlows

ClowdFlows is a cloud-based web application for distributed computing that supports the construction and execution of data mining workflows, including web services used as workflow components. Big data analytics is provided through several algorithms and novel ensemble techniques are supported. ClowdFlows has a service-oriented architecture and can also function as a real-time stream mining platform with an intuitive user interface. ClowdFlows aims to address the challenges faced in managing the lifecycle of machine learning models by providing a comprehensive platform. Its architecture encompasses multiple stages of ML workflows, enabling efficient data pre-processing, model training, deployment, and monitoring.(16)(17) Figure 4 provides an overview of the architecture.

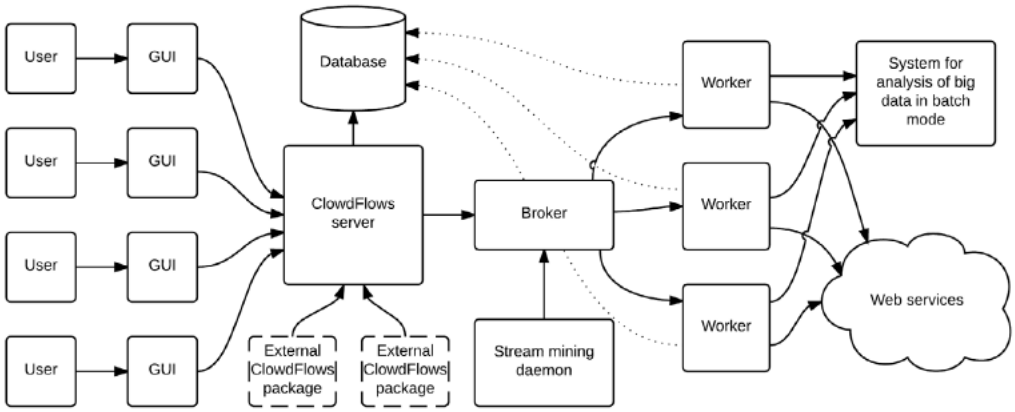


Figure 4: ClowdFlows architecture

Architecture The architecture of ClowdFlows(17) comprises the following key components:

Graphical user interface: The GUI is developed with HTML, and jQuery and is supplied from the central ClowdFlows server. It allows users to add, delete, reposition, and connect multiple components to form a unified workflow. It is in charge of displaying results in an understandable way as well as displaying a library of publicly available workflows that users can duplicate. The code is executed within users' web browsers, resulting in a responsive and interactive experience.

ClowdFlows Server: ClowdFlows server's software is built in Python and on the Django web platform. It is made up of two parts: a web application and a widget repository. Models, views, and templates are defined by the web application. The data model is an abstract representation of workflows and widgets. A widget is a workflow processing unit that includes inputs, outputs, and parameters and is interconnected by 'Connections'.

Database: The workflows and all user-uploaded data are stored in the database. Django's object-relational mapper provides an API for connecting objects to databases, rendering the platform database agnostic.

Broker: The broker ensures that the tasks are evenly distributed across the worker nodes.

Worker nodes: Worker instances are ClowdFlows server instances that can only be accessed by the broker. They execute workflows and workflow components. The workers send success or error messages to the broker and have timeouts to ensure fault tolerance if a worker goes down during runtime.

Stream mining daemon: The stream mining daemon is a process that runs alongside the ClowdFlows server, loops through deployed stream mining workflows, and executes them.

Web Services: The PySimpleSoap module is used to facilitate web service consumption and import as workflow components.

Features Various features of ClowdFlows were discussed at length.(17) (18) Description of its key features are as follows:
Public workflows: Users are allowed to create public versions of their workflows by generating a

URL and publicizing it. When accessed, a copy is created and added to the user's private workflow repository.

Widget Development: A widget can be written as a Python function and included in a ClowdFlows package or can be manually imported as a WSDL Web service via the graphical user interface, thus supporting web services.

Scalability: The ClowdFlows platform can be scaled horizontally by scaling the database, broker, and worker nodes. ClowdFlows server can be scaled by installing it on numerous machines and running it behind a load-balancing web server like 'Nginx'. If the data passing from one workflow component to another is considerable, the broker must be scaled. Increasing the number of workers is the most common scaling operation, as each worker can execute a certain number of widgets concurrently at any given time.

Real-time data stream mining: ClowdFlows enables real-time data stream processing by using a stream mining daemon and a modified workflow execution engine with a halting mechanism. Results showed that by using a single worker, data can be processed three times slower than the rate of production on the input and preserve the same throughput. Using two workers, data can be processed ten times slower, while a configuration with three workers allows for a twenty times slower processing rate. This allows users to have complex workflows perform analyses on the data and still see results in real-time while being confident that all data was processed.(19)

Batch Data Processing: Disco MapReduce framework was chosen to perform MapReduce tasks, as it is written in Python and allows for easier integration with the ClowdFlows platform. However, there is a lack of a specialized machine learning library or toolkit within the framework, leading to the development of ClowdFlow's own library with a limited set of machine learning algorithms, 'DiscoMLL'.

Support of distributed ensemble methods: ClowdFlows developed and adapted tree-based ensembles such as 'Forest of Distributed Decision Trees', 'Distributed Random Forest' and 'Distributed Weighted Forest' for distributed computation with MapReduce.

Strengths The paper provides technical details about the implementation and capabilities of ClowdFlows and describes how the platform supports web services, integrates the Disco framework for distributed computing, and includes a machine learning library for batch processing of big data. It also presents use cases and evaluates the platform's performance. The use cases are comprehensive and end-to-end, also acting as specialized tutorials of the platform. ClowdFlows is released under an open-source license and is publicly available on the web. This availability encourages wider adoption and allows users to customize and deploy the platform according to their specific needs.

Weaknesses Although the study mentions the comparison of ClowdFlows with related platforms, it does not provide an in-depth analysis or evaluation of these comparisons. More detailed comparisons and discussions would have provided a clearer understanding of the unique advantages of ClowdFlows in the context of existing solutions. The study also highlights the strengths and advanced features of ClowdFlows, but it does not explicitly address any limitations or potential challenges associated with the platform. A more balanced discussion that acknowledges potential drawbacks or areas for improvement would have enhanced the paper's credibility.

3.5 Distributed GraphLab

Distributed GraphLab or now called GraphLab Create is a cloud-based machine learning platform. It is designed to simplify the development of intelligent applications by providing a high-level interface for performing various machine-learning tasks. GraphLab create provides features like built-in algorithms, tools for data preprocessing and feature engineering, flexible data structures, and model deployment. It is also designed for horizontal scaling across multiple machines to perform distributed computing efficiently.

In this paper(20) (21), the author highlights the limitations of the existing large-scale frameworks, the implementation of GraphLab Abstraction, two methods of implementing a new distributed execution model, fault tolerance of the model, implementation of three state-of-the-art models with GraphLab abstraction, evaluation of Graphlab using 512 processor EC2 server.

Architecture The GraphLab abstraction consists of three main parts, the data graph, the update function, and the sync operation. The data graph stores the users' modifiable state in a data graph. For example, the data graph is derived directly from the web graph, wherein each vertex corresponds to a web page and each edge represents a link. The vertex data D_v holds R(v), which is the current estimation of the PageRank, while the edge data $D_{u \to v}$ stores u,v, indicating the directed weight of the link. The update function is a stateless procedure to modify the data within the scope of the vertex. The scope of vertex v, represented as S_v, encompasses the data stored within v itself, as well as the data stored in all neighboring vertices and adjacent edges.

The GraphLab abstraction offers a comprehensive sequential model that is automatically transformed into parallel execution by enabling multiple processors to execute the same loop on the same graph. To preserve the sequential execution semantics, various consistency models have been introduced to optimize parallel execution while ensuring serializability. Additionally, global statistics that describe the data contained in the data graph must be preserved, so the GraphLab abstraction introduces global values that can be accessed by update functions, but their modification is performed using sync operations. Sync operations are associative commutative sums defined overall by the scopes in the graph.

The authors describe a two-phased partitioning-based distributed data graph model that enables effective loading on a variety of Cloud deployments. They encode connection structure and file locations in a meta-graph, allowing each machine to build its own local portion of the graph. Two engine variants are discussed: the chromatic engine, which achieves parallel execution using vertex coloring, and the distributed locking engine, which extends the mutual exclusion technique for increased scheduling flexibility. To reduce latency, a number of strategies are used, including caching, pipelined locking, and prefetching. On a large-scale graph, the distributed pipelining system's performance is assessed, showing a notable decrease in runtime.

Fault tolerance is introduced to the distributed GraphLab framework through a distributed checkpoint mechanism. Two strategies for constructing distributed snapshots are evaluated: synchronous and fully asynchronous. The ideal checkpoint interval must strike a compromise between the expense of building a checkpoint and the computation lost since the last checkpoint. Performance evaluation of the snapshotting algorithms on a synthetic mesh problem shows that the asynchronous snapshot has minimal impact on execution speed, while the synchronous snapshot halts execution. In a multi-tenancy environment, asynchronous snapshots' benefits are more obvious.

GraphLab uses a Map-Reduce process to create an atom graph representation on a Distributed File System (DFS). It uses an asynchronous RPC protocol via TCP/IP to communicate between processes running on several machines. A cache is used to access remote graph data, and a scheduler manages the assigned vertices in a group of processes named T. To detect when each scheduler has emptied its sets, a distributed consensus approach is used.

Evaluation GraphLab was evaluated on three MLDM applications: Netflix movie recommendations, Video Co-segmentation, and Named Entity Recognition (NER). Performance experiments were conducted on Amazon's Elastic Computing Cloud (EC2) using up to 64 High-Performance Cluster instances. On comparable jobs, GraphLab outperformed Hadoop 20–60 times faster and demonstrated performance similar to specialized MPI implementations, with scaling becoming more efficient at larger computation–to–communication ratios. In the Netflix movie recommendation assignment, GraphLab outperformed Hadoop and MPI solutions in terms of accuracy and performance. With more machines and a longer pipeline, GraphLab's locking engine was able to significantly speed up Video Co-segmentation. The performance boost of GraphLab in the NER application, however, was only moderate and reached saturation after 16 computers, showing the need for more improvements to improve scalability and match the performance of dedicated MPI implementations.

Strengths The paper highlights the increasing need for efficient parallel execution of MLDM algorithms on large clusters due to the exponential growth in the scale of MLDM problems. It also acknowledges the challenges involved in designing, implementing, and debugging distributed MLDM algorithms, which require expertise in both MLDM and parallel/distributed computing. It emphasizes the need for a high-level distributed abstraction that targets the asynchronous, dynamic, graph-parallel computation found in many MLDM applications while hiding the complexities of parallel/distributed system design.

Weaknesses The paper largely contrasts GraphLab's implementations with those of Hadoop, Pregel, and MPI. While these comparisons are useful, a more thorough evaluation would come from a wider comparison with other distributed MLDM systems or frameworks. Additionally, a special configuration of a 512-processor (64-node) cluster is used for the Amazon EC2 platform during the evaluation of Distributed GraphLab. In alternative deployment situations or with different hardware configurations, the performance and scalability of Distributed GraphLab are not fully investigated. Furthermore, the paper concentrates on particular MLDM methods, which could not accurately reflect Distributed GraphLab's performance and suitability for a variety of MLDM applications. Further research should be done to determine whether the results are generalizable to other MLDM algorithms or problem domains. Also, while the paper mentions the incorporation of fault tolerance through snapshotting schemes, it provides limited discussion or evaluation of the fault tolerance mechanisms. The effectiveness and efficiency of fault tolerance methods might be further investigated and tested, which would improve our understanding of their overall impact.

4 Discussion & Conclusion

The management of resources in machine learning (ML) pipelines is a complex task that requires careful consideration of various stages and their corresponding resource requirements. Under-provisioning or over-provisioning can lead to inefficiencies and sub-optimal performance. To address these challenges, several ML frameworks have been developed, each offering unique features and capabilities.

CIRRUS leverages the serverless infrastructure provided by AWS Lambda and overcomes resource constraints by acting as a wrapper around Lambdas. DEEP focuses on enabling distributed training and sharing of ML models, emphasizing model portability and collaboration. Kubeflow, built on Kubernetes, offers a comprehensive set of tools for ML model development and management, providing scalability and portability. ClowdFlows is a cloud-based web application that supports the construction and execution of data mining workflows. However, the versions of the platforms discussed in this paper were produced after many revisions made by the developers and/or authors. These platforms are also ever-evolving to handle the problems that arise with the explosion of data.

Numerous opportunities for future work and research exist because of how quickly the fields of ML and MLOps are developing. Future research can concentrate on further analyzing the functionality and scalability of MLOps tools. Comparative research might be done to evaluate how well various tools handle massive ML pipelines and datasets. In order to enable smooth workflow management, there is also room to improve these tools' ability to integrate with well-known ML frameworks and libraries like TensorFlow or PyTorch. It may also be focused on creating MLOps frameworks that explicitly address edge deployment scenarios, taking into account the growing popularity of edge computing and the requirement for ML models to be deployed on resource-constrained devices. These instruments might help with issues like model compression, device optimization for low power consumption, and effective model updates in edge contexts.

Furthermore, it is evident that while there are a huge number of MLOps tools present in the market, no tool is yet able to provide end-to-end support for non-thematic machines or deep learning pipelines. While some only support data processing and model processing, others support model processing and result visualization. Scientists and developers often have to switch between tools for their desired tasks.

To sum up, ML/DL frameworks are constantly developing to solve the issues and challenges faced in ML pipelines. We have learned about the advantages and limitations of different ML/DL technologies through research and analysis. The effectiveness, scalability, and adaptability of these technologies in managing ML workflows can still be greatly improved with further work and research.

References

[1] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A. and Suter, P., 2017. Serverless computing: Current trends and open problems. Research advances in cloud computing, pp.1-20.

[2] Kreuzberger, D., Kühl, N. and Hirschl, S., 2023. Machine learning operations (MLOps): Overview, definition, and architecture. IEEE Access.

[3] Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I., 2010. Spark: Cluster computing with working sets. HotCloud, 10(10-10), p.95.

[4] García, Á.L., De Lucas, J.M., Antonacci, M., Zu Castell, W., David, M., Hardt, M., Iglesias, L.L., Moltó, G., Plociennik, M., Tran, V. and Alic, A.S., 2020. A cloud-based framework for machine learning workloads and applications. IEEE access, 8, pp.18681-18692.

[5] Zhou, Y., Yu, Y. and Ding, B., 2020, October. Towards MLOps: A case study of ML pipeline platform. In 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE) (pp. 494-500). IEEE.

[6] Ruf, P., Madan, M., Reich, C. and Ould-Abdeslam, D., 2021. Demystifying MLOps and presenting a recipe for the selection of open-source tools. Applied Sciences, 11(19), p.8861.

[7] Continuous delivery for machine learning. https://martinfowler.com/articles/cd4ml.html

[8] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F. and Dennison, D., 2015. Hidden technical debt in machine learning systems. Advances in neural information processing systems, 28.

[9] Carreira, J., Fonseca, P., Tumanov, A., Zhang, A. and Katz, R., 2019, November. Cirrus: A serverless framework for end-to-end ml workflows. In Proceedings of the ACM Symposium on Cloud Computing (pp. 13-24).

[10] http://pywren.io/

[11] https://www.kubeflow.org/docs/started/architecture/

[12] https://www.run.ai/guides/kubernetes-architecture/kubeflow-pipelines-the-basics-and-a-quick-tutorial

[13] https://www.kubeflow.org/docs/started/introduction/

[14] https://aws.amazon.com/blogs/machine-learning/build-flexible-and-scalable-distributed-training-architectures-using-kubeflow-on-aws-and-amazon-sagemaker/

[15] Pandey, A., Sonawane, M., & Mamtani, S. (2022). Deployment of ML Models using Kubeflow on Different Cloud Providers. ArXiv, abs/2206.13655.

[16] Kranjc, J., Podpečan, V., Lavrač, N. (2012). ClowdFlows: A Cloud Based Scientific Workflow Platform. In: Flach, P.A., De Bie, T., Cristianini, N. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2012. Lecture Notes in Computer Science(), vol 7524. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33486-3_54

[17] Kranjc, J., Orač, R., Podpečan, V., Lavrač, N., Robnik-Šikonja, M., 2017. ClowdFlows: Online workflows for distributed big data mining, Future Generation Computer Systems, Volume 68, Pages 38-58, ISSN 0167-739X, https://doi.org/10.1016/j.future.2016.07.018.

[18] Kranjc, J., Smailović, J., Podpečan, V., Grčar, M., Žnidaršič, M., Lavrač, N., 2015. Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform, Information Processing & Management, Volume 51, Issue 2, Pages 187-203, ISSN 0306-4573, https://doi.org/10.1016/j.ipm.2014.04.001

[19] J. Kranjc, V. Podpečan, N. Lavrač, "Real-time data analysis in ClowdFlows," 2013 IEEE International Conference on Big Data, Silicon Valley, CA, USA, 2013, pp. 15-22, doi: 10.1109/BigData.2013.6691682.

[20] Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., Hellerstein, J., 2012. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. Proceedings of the VLDB Endowment. 5. 10.14778/2212351.2212354.

[21] Low, Y., Gonzalez, J.E., Kyrola, A., Bickson, D., Guestrin, C., & Hellerstein, J.M. (2010). GraphLab: A New Framework For Parallel Machine Learning. ArXiv, abs/1006.4990.

OAuth2, Webhooks, API in micro-services

Rohaan Almeida
Najma Christen
Ingrid Marie Wølneberg
Department of Computer Science
University of Amsterdam

June 5, 2023

Abstract

This paper explores three independent directions OAuth 2.0, webhooks, and API management, and their impact on secure and efficient API integration. We explore into the core concepts, benefits, and challenges associated with each of these technologies, we aim to provide a comprehensive understanding of their role in modern application integration. Furthermore, we will discuss real-world use cases and best practices. We also compare these technologies to similar technologies used in the industry as well as present the latest improvements in these domains.

1 Introduction

In today's digital landscape, where interconnectivity and data exchange are paramount, organizations rely on robust and efficient methods to integrate their applications, services, and systems seamlessly. OAuth 2.0, webhooks, and API management have emerged as critical components in this integration ecosystem, providing secure, scalable, and standardized mechanisms for data sharing and interaction.

2 Research Question

What are the identified research directions concerning communication and deployment of microservices with respect in areas such as Oauth2, webhooks and API Integration?

1. RQ1: What are good API management standards when it comes to microservice architecture?
2. RQ2: How does OAuth2 compare to other authentication and authorization protocols, such as SAML and OpenID Connect?

3. RQ3: What are the advantages of webhooks compared to the polling practice in microservices?

3 API management

3.1 API management practices for microservices

In a microservice architecture the services communicate with each other through APIs, in addition to this APIs can be used for communication with external third party systems. This makes management of those APIs important, as it is key to efficient communication. Using clever API management solutions for your architecture can enable policy and key validation, service versioning, quota management, authorizations and access control etc. [IS18].

In the book "Microservices for the Enterprise" ch.10: "API, Event, and Streams" they identify six components of an API management solution. These components are called API publisher, API Developer Portal/Store, API Gateway, API Analytics/Observability, API Monetization and API Quality of Service. So there are several management practices that have emerged to address requirements and challenges of microservice architecture. API management handles the management through the whole API lifecycle from design to maintainment and retirement. In some cases API Gateway is used interchangibily with API Management because the API Gateway is a crucial part of the management [IS18].

Another paper discusses the importance of API gateways in the microservices architecture, highlighting their role as the entry point for requests and their ability to simplify communication between clients and microservices. It explores key functionalities such as authentication, reverse proxy, and flow control, using technologies like OpenResty, Nginx, and Lua. The paper emphasizes the significance of continuous updates and optimization to ensure high performance and efficient development in managing microservice interfaces [ZJJ18].

We found a Systematic Literature Review that highlight the importance of API Management in general. Their goal is to identify practices and capabilites that can be found in literature, as well as creating a new definition for API Management based on the different definitions they encountered in their review. Even if this review is not specifically related to microservices, the same practices would apply and I would argue that API Management is even more important for a microservice architecture as communication internally is also based on APIs [MOJ20]. The authors identify 8 practices related to API Management: caching, OAuth authentication, load balancing, rate/quota limiting, usage throttling, activity logging/monitoring, access control and billing [MOJ20].

3.2 What are identified challenges within API management and are there any suggested solutions?

Since the API gateway is now a central part of the microservice architecture, there are some challenges related to this component. It exists many API gateways developed by large enterprises over the years. It is possible for companies to buy some of these API gateways and some are even open source. However for many it can be too expensive buying an API gateway solution and too time consuming and difficult to configure open source code to meet their own systems requirements. Another approach could then be to use Nginx to deploy their own API Gateway but this solution can also be difficult to develop to work as well as the commercial API Gateways [ZSWX20]. So it seems that one important challenge when it comes to API Gateways is how to choose the right one for your system. This paper suggest a solution to this cost, performance and maintainability issue of choosing API gateway for your system. The authors suggests to optimize the API Gateway by optimizing the framework of the API Gateway, reducing coupling between the API Gateway core business and plug-ins. After finalizing the optimization they compare it with an open source based API Gateway and the Nginx solution, and their result do indeed show that their suggested solution improve response rate and performance of the API Gateway [ZSWX20].

While it can be difficult to find the best API Gateway solution for your system there are also some general challenges when it comes to API Management. We found this in different tech blogs or websites and from that tried to find scientific papers who might be addressing these problems. Some of the challenges we found were [Kon23, NGI22, mer, Wen23]:

- Gateway specific challenges:
  - The gateway is an extra component that would need to be developed and maintained/updated.
  - An extra network hop for all the requests to your system.
  - Potential bottleneck or single point of failure.
- Documentation
- Visibility and governance
- Versioning: Akbulut and Perros propose a solution to the versioning problem by extending and modifying the API gateway pattern. According to their paper their new approach runs the system with 27% less hosting cost compared to a normal auto-scaling method [AP19].
- Security: a solution to this is discussed further in the next section on OAuth2.

4 OAuth2

4.0.1 Why is Authorization(Oauth2) preferred over authentication for API Communication?

When comparing authorization to authentication in terms of performance, authorization generally has some advantages:

1. **Reduced Authentication Overhead**: Once a user or client has been authenticated and a valid session or token is available, further authorization checks can be performed without the need for re-authentication, thus significantly reducing the computational and network overhead associated with authentication, improving overall performance.
2. **Caching and Access Control Lists (ACLs):** Caching allows frequently accessed authorization decisions to be stored in memory, thus reducing the need to perform redundant authorization checks repeatedly. This caching mechanism can improve performance by providing faster responses for subsequent authorization requests.
3. **Efficient Resource Access:** By defining specific access rules or permissions, authorization mechanisms can optimize the access process and limit unnecessary requests to resources.
4. **Easier Scalability:** By employing scalable authorization models, we can ensure that access control processes can handle the increasing number of concurrent requests without impacting performance.
5. **Reduced Network Latency:** Authorization decisions can often be made amongst close neighbors to the resource being accessed, thus reducing the need for network round trips or through using lightweight tokens, the overall network latency can be minimized, leading to better performance and response times.
6. **Role-Based Access Control (RBAC):** RBAC maps users or entities to predefined roles, and authorization decisions are based on the roles assigned to them. This simplification reduces the complexity of the access control process and can lead to faster and more efficient authorization checks.[CP22, TAS$^+$22, ACC$^+$08, Ngu, Yan, MRS$^+$23]

4.0.2 How does Oauth2 compare against other protocols such as SAML and OpenID?

Here are the key differences between SAML, OpenID and OAuth 2.0:

1. Purpose and Scope:
    - **SAML**: SAML primarily deals with authentication and exchanging identity-related information between an identity provider (IdP) and a service provider (SP). It enables SSO and allows users to authenticate once and access multiple applications without re-authentication.

- **OpenID**: OpenID focuses on user authentication and provides a way for users to prove their identity to multiple websites or applications using a single set of credentials. It allows users to authenticate themselves through an identity provider (IdP) and obtain an identity token that can be presented to service providers (SP) for authentication.
- **OAuth 2.0**: OAuth 2.0 focuses on authorization and access delegation. It enables secure access to protected resources on behalf of a user. OAuth 2.0 is commonly used in scenarios where a user wants to grant limited access to their resources to a third-party application.

2. Protocol:

   - **SAML**: SAML uses XML-based messages for communication between the IdP and SP. It follows a request-response model where the user is redirected from the SP to the IdP for authentication, and after successful authentication, the IdP sends an assertion (SAML token) back to the SP.
   - **OpenID**: OpenID relies on the use of protocols such as OpenID Connect (built on top of OAuth 2.0) and the use of identity tokens. It provides an authentication layer that uses OAuth 2.0 as the underlying authorization framework.
   - **OAuth 2.0**: OAuth 2.0 primarily relies on HTTP-based protocols for communication. It utilizes a token-based approach, where the client (third-party application) obtains an access token from an authorization server to access protected resources on the resource server through signature verification.

3. Flow and Parties Involved:

   - **SAML**: The SAML flow typically involves three parties: the user (resource owner), the IdP, and the SP. The user authenticates with the IdP, which generates a SAML token containing identity assertions. The user then presents this token to the SP to access the desired application or resource.
   - **OpenID**: OpenID involves three main parties: the user, the identity provider (IdP), and the relying party (RP). The user authenticates with the IdP and receives an identity token. The user can then present this token to the RP to authenticate themselves and gain access to the desired application or service.
   - **OAuth 2.0**: The OAuth 2.0 flow involves four parties: the user (resource owner), the client (third-party application), the authorization server, and the resource server. The user grants permission to the client to access their resources by interacting with the authorization server. The client then obtains an access token from the authorization server, which it uses to access protected resources on the resource server.

4. Use Cases:

    - **SAML**: SAML is commonly used in enterprise environments and is well-suited for scenarios where a user needs to access multiple applications with a single set of credentials, such as SSO across different systems or federated identity management.
    - **OpenID**: OpenID is commonly used for scenarios where users want to have a single sign-on experience across multiple websites or applications, using a single set of credentials. It provides a convenient and secure way to authenticate users and manage their identities.
    - **OAuth 2.0**: OAuth 2.0 is widely used in web and mobile applications that require secure access to user resources, such as accessing APIs or fetching data from social media platforms.

In summary, SAML, OpenID and OAuth 2.0 serve different purposes. SAML focuses on identity management and SSO, OpenID focuses on user authentication and identity verification, providing a way for users to authenticate across multiple websites or applications whereas OAuth 2.0 focuses on authorization and access delegation. All three protocols play important roles in enabling secure and controlled access to resources, but they have different scopes and use cases. [Hug, MRS+23, ACC+08, SBJ+14]

4.0.3 What are some alternatives to Oauth2 that that offer additional features or improvements?

1. **OAuth2.1**: OAuth2.1 is an evolution of OAuth2.0 that aims to address some of the security and usability concerns that have emerged over time. It provides clarifications, updates, and best practices to enhance the security posture of OAuth2.0 implementations such as secure communication, token storage, token revocation, and token expiration.

2. **FIDO2**: Fast Identity Online 2 (FIDO2) is an authentication protocol that goes beyond traditional username/password-based authentication. FIDO2 uses public key cryptography and hardware-backed security to enable passwordless and strong two-factor authentication. It provides a more secure and user-friendly authentication experience compared to OAuth2.0 alone.

3. **SCIM**: System for Cross-domain Identity Management (SCIM) is not an authentication or authorization protocol itself but rather a standard for user provisioning and management. It defines a schema and RESTful APIs for managing user identities across different systems. SCIM can be used in conjunction with OAuth2.0 or other protocols to facilitate user provisioning and identity synchronization. [NJ17, LSN+20, FLS+20, Nad13]

5 Webhooks

In today's world effective and smooth communication between applications and services has become essential. Webhooks is a powerful tool for real-time data exchange and turns out to be a better in many ways than traditional polling methods. In the following we will explore crucial differences and in particular advantages of webhooks compared to polling.

- **Real-Time Communication**: Webhooks facilitate real-time communication between applications by instantly notifying receiving systems about specific events or changes. Unlike polling, which involves continuously querying for updates, webhooks eliminate the need for constant requests. With webhooks, relevant data is delivered instantly when an event occurs, enabling immediate reactions and reducing latency in communication. This real-time aspect ensures that information is up-to-date and allows systems to respond promptly, enhancing the overall efficiency and responsiveness of applications. The comparative study by Kavats [KK19] on a telegram server shows that Webhooks performs up to three times better than standard polling on an increasing number of requests per second.

- **Resource Efficiency**: Polling involves making frequent requests to a server, consuming resources and increasing server load, even when there is no new data available. In contrast, webhooks eliminate unnecessary requests, significantly reducing resource consumption. With webhooks, data is transmitted only when an event occurs, minimizing bandwidth usage and freeing up server resources. This efficiency not only optimizes system performance but also reduces operational costs, making webhooks an attractive solution for resource-constrained environments. Duner's study [DN20] suggests, that webhooks can handle a higher load than polling before reaching the CPU limits.

- **Event-Driven Architecture**: Webhooks follow an event-driven architecture, enabling decoupling between systems. Instead of relying on continuous polling, applications can subscribe to specific events and receive notifications only when relevant data is available. This decoupling encourges modular design, allowing independent development and scalability of individual components. Moreover, the event-driven approach enables greater flexibility, as systems can dynamically subscribe or unsubscribe to events based on their specific requirements. This supports adaptability and extensibility, as it is easy for systems to adjust to changing requirements. It also makes the system more robust as the services work independently, the failure of one service does not interfere with the rest of the services.[Var23]

- **Improved Data Integrity**: Webhooks ensure data integrity by delivering information in the most reliable way. Unlike polling, where data might be subject to synchronization issues due to time delays, webhooks eliminate such concerns. When an event occurs, webhooks deliver the

data immediately, reducing the risk of data inconsistency. This advantage is particularly crucial in scenarios where data accuracy and consistency are extremely important, such as financial transactions or collaborative applications.

- **Simplified Integration**: Webhooks simplify the integration process between different systems. By adhering to standard HTTP protocols and leveraging familiar request/response mechanisms, webhooks can be easily integrated into existing infrastructure. Developers can utilize widely supported web frameworks and libraries to handle incoming webhook requests, reducing implementation complexity. This simplicity of integration promotes faster development cycles and facilitates seamless communication across diverse applications.

6 Discussion

6.1 Difficult to compare

When it comes to comparing different architectures, such as API management architectures, it can be challenging due to their varying scopes and use cases. Each architecture is designed to address specific requirements and objectives.

6.2 Future improvements and research

The systematic literature review [MOJ20] suggest some future work in the area of API Management which we can also be a possible solution to the comparison issue. They suggest that to enable organizations to evaluate and improve their API management maturity levels, a Future Area Maturity Model (FAMM) can be built. We also think this model can be used to more easily compare API Management solutions with each other as it will make parameters for comparison more clear. However this still requires more research into the practices, capabilities and the relationships between them. As well as verification from experts.

However this literature review is not focused only on microservice architecture, just API Management in general. We therefore wonder if there might be some differences in the "best practices" for API Management depending on what type of architecture your system is using.

Future improvements to Oaut2.0 is Oauth 2.1 and it aims to address some security and usability concerns. Additionally System for Cross-domain Identity Management (SCIM) be used with OAuth2.0 or other protocols to facilitate user provisioning and identity synchronization.

7 Conclusion

In this literature study, we inspected three different directions in API integration, which all have a connection to microservices in a different way.

API management practices, including API gateways, versioning, authentication, and access control, are essential for facilitating smooth communication both internally and with external systems. Choosing the right API gateway solution and optimizing its performance are important challenges to address.

OAuth 2.0, applying a token-based approach, stands out as a preferred authorization protocol in microservices due to its reduced authentication overhead, caching capabilities, efficient resource access, scalability, and reduced network latency. While SAML and OpenID also serve their purposes in authentication and identity management, OAuth2 specifically focuses on authorization and access delegation.

Webhooks offer significant advantages over polling practices in microservices. Real-time communication, resource efficiency, event-driven architecture, improved data integrity, and simplified integration make webhooks a superior choice for communication in microservices.

These technologies and practices contribute to enhanced performance, scalability, and adaptability, supporting the successful implementation and management of microservices. Hence, it can be of high importance to consider OAuth 2.0, webhooks and a proper API management to achieve a robust and efficient microservice.

References

[ACC+08] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra. Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering*, FMSE '08, page 1–10, New York, NY, USA, 2008. Association for Computing Machinery.

[AP19] Akhan Akbulut and Harry G Perros. Software versioning with microservices through the api gateway design pattern. In *2019 9th International Conference on Advanced Computer Information Technologies (ACIT)*, pages 289–292. IEEE, 2019.

[CP22] Ayan Chatterjee and Andreas Prinz. Applying spring security framework with keycloak-based oauth2 to protect microservice architecture apis: A case study, Feb 2022.

[DN20] Daniel Dunér and Marcus Nilsson. Scalability of push and pull based event notification: A comparison between webhooks and polling, 2020.

[FLS+20] Florian M Farke, Lennart Lorenz, Theodor Schnitzler, Philipp Markert, and Markus Dürmuth. " you still use the password after all"—exploring fido2 security keys in a small company. In *Proceedings of the Sixteenth USENIX Conference on Usable Privacy and Security*, pages 19–35, 2020.

[Hug] J Hughes. Citeseerx.

[IS18] Kasun Indrasiri and Prabath Siriwardena. *APIs, Events, and Streams*, pages 293–312. Apress, Berkeley, CA, 2018.

[KK19] Elena Aleksandrovna Kavats and Artem Aleksandrovich Kostenko. Analysis of connection methods of telegram robots with server part. *System technologies*, 3(122):19–24, 2019.

[Kon23] Why do microservices need an api gateway, Apr 2023.

[LSN+20] Sanam Ghorbani Lyastani, Michael Schilling, Michaela Neumayr, Michael Backes, and Sven Bugiel. Is fido2 the kingslayer of user authentication? a comparative usability study of fido2 passwordless authentication. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 268–285. IEEE, 2020.

[mer] 7 api management challenges (and how to solve them).

[MOJ20] Max Mathijssen, Michiel Overeem, and Slinger Jansen. Identification of practices and capabilities in api management: a systematic literature review. *arXiv preprint arXiv:2006.10481*, 2020.

[MRS+23] Gunasekaran Manogaran, Bharat S. Rawal, Vijayalakshmi Saravanan, Priyan M K, Qin Xin, and P. Shakeel. Token-based authorization and authentication for secure internet of vehicles communication. *ACM Trans. Internet Technol.*, 22(4), mar 2023.

[Nad13] T Nadalin. Oauth working group p. hunt, ed. internet-draft oracle corporation intended status: Standards track m. ansari expires: January 06, 2014 cisco. 2013.

[NGI22] Building microservices: Using an api gateway, Dec 2022.

[Ngu] Quy Nguyen. Applying spring security framework and oauth2 to protect microservice architecture api.

[NJ17] Nitin Naik and Paul Jenkins. Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 163–174. IEEE, 2017.

[SBJ+14] Natsuhiko Sakimura, John Bradley, Mike Jones, Breno De Medeiros, and Chuck Mortimore. Openid connect core 1.0. *The OpenID Foundation*, page S3, 2014.

[TAS+22] Michal Trnka, Amr S. Abdelfattah, Aishwarya Shrestha, Michael Coffey, and Tomas Cerny. Systematic review of authentication and authorization advancements for the internet of things. *Sensors*, 22(4), 2022.

[Var23] Thomas Cole Varney. Webhooks-as-a-service: A custom api design. 2023.

[Wen23] Mark Wentowski. Api lifecycle management: Phases, challenges amp; best practices, May 2023.

[Yan] Ronghai Yang. Signing into one billion mobile app accounts effortlessly with oauth2.

[ZJJ18] JT Zhao, SY Jing, and LZ Jiang. Management of api gateway based on micro-service architecture. In *Journal of Physics: Conference Series*, volume 1087, page 032032. IOP Publishing, 2018.

[ZSWX20] Xianyu Zuo, Yuehan Su, Qianqian Wang, and Yi Xie. An api gateway design strategy optimized for persistence and coupling. *Advances in Engineering Software*, 148:102878, 2020.

MicroVMs and unikernels for the cloud

Web Services and Cloud-Based Systems - Group 33

Niclas Haderer - UvA Amsterdam
14340208
niclas.haderer@student.uva.nl

Paulo Liedtke - UvA Amsterdam
14209772
paulo.liedtke@student.uva.nl

Paul Groß - UvA Amsterdam
14723700
paul.gros@student.uva.nl

June 4, 2023

Abstract

This paper examines the trade-offs between microVMs, unikernels, and containers in terms of resource consumption, performance, security, usability, and tooling. The study concludes that while containers maintain a performance advantage, certain configurations of microVMs in combination with unikernels can outperform them in specific scenarios. Despite their higher isolation potential, unikernels suffer from missing several kernel security features and have challenges because of their complex build process and lack of robust community support. However, the integration of microVMs with Kubernetes is starting to address these issues. It is concluded that microVMs and unikernels serve as promising alternatives to containers for large-scale, high-performance projects requiring higher isolation where the extra implementation time is worth a bit more performance and lower startup times. This e.g. is the case when working under constrained resource conditions such as edge computing. MicroVMs with a small Linux kernel are more suitable when higher isolation is necessary, as seen in AWS Lambda functions, allowing for optimized hardware utilization.

1 Introduction

The prevalence of containers for software deployment in cloud environments is widely acknowledged [41]. However, especially for security critical environments they have problems. This is amongst other things due to an overly exposed host kernel that the containers have to interact with [27, p. 219]. This is a potential surface for attacks [33, 38], which is hard to secure [15] and has been exploited multiple times [14, 15]. This raises concerns for public cloud providers because they have limited control over the kind of applications running on their cloud. Because of that, cloud service providers like Amazon and Google shifted to microVMs for some of their cloud products, like lambda functions [1]. This is because microVMs offer higher isolation compared to containers [1]. In contrast to classical VMs like VMWare, containers have extremely fast instantiation times, small memory footprints, and high density on the host system [27, p. 219].

The goal of the paper is to compare the trade-offs between microVMs, unikernels, and Containers in terms of resource consumption, performance, security, and manageability to answer the question of if and where new technologies like microVMs and unikernels are viable alternatives for containers.

1.1 Theoretical background

1.1.1 Containers

Containers are lightweight, isolated runtime environments that contain an application and all its dependencies. Normally a container only contains a single application. Because containers do not run in a hypervisor low-level calls like system calls are forwarded to the host operating system [40, 27]. The isolation is based on cgroups (Control Groups) which can be used to limit and isolate a process's access to other CPUs, RAM, and networks. Linux namespaces can then be used to create a protected environment for every running container [6]. Because docker containers do not include a kernel their resource usage is relatively low and startup speeds are generally very fast [23, 27].

1.1.2 MicroVMs

A microVM is a highly specialized virtual machine designed to execute specific types of applications, supporting only basic capabilities [27]. Through simplification, the elimination of unnecessary devices, and the removal of unnecessary functionalities, a microVM achieves quicker startup times and lower resource utilization when compared to a traditional VM [27] [38]. Because the kernel is virtualized and microVMs provide hardware-level isolation a microVM offers more isolation and therefore also security than a container [27]. Furthermore, a microVM can enforce limitations on CPU utilization, memory allocation, and network usage. This offers protection from DOS (denial of service) attacks and prevents a single VM from using all resources [38].

1.1.3 Unikernels

Unikernels are images intended to run only a single application at the same time [41]. Once deployed, unikernels cannot be modified [25]. They contain all the application code, libraries, and OS/Kernel within a unified address space [25]. Because they are compile-time linked it is possible to eliminate unnecessary components thereby reducing the attack surface by shipping fewer libraries and decreasing the size of the resulting image [20]. Due to their highly specialized nature and customizability to meet specific application requirements, unikernels provide access to low-level APIs, resulting in superior performance compared to traditional operating systems [41] in some scenarios. However because of their minimalist nature some security features, which are normally found in an OS might be missing. Furthermore, unikernels also introduce certain challenges, such as making the debugging of deployed applications more complex and being more difficult to build compared to Docker containers[9]. The resulting unikernel can be deployed either on a hypervisor or on bare metal [26].

1.2 Methodology

In order to compare microVMs, unikernels, and containers a systematic literature review according to the definition of Snyder [35] was performed. Therefore, relevant literature consists of studies comparing papers on microVMs, unikernels, and containers. To identify relevant literature, the abstract and citation database Scopus was used [10]. Scopus, as per Kulkarni et al.'s findings, outperforms its competitor Web of Science in retrieving citations with comparable performance to Google Scholar, while also providing a more capable search functionality than Google Scholar [21]. The search query aimed to find English literature published from 2015 onward that includes the keywords 'microVM(s)' or 'unikernel(s)' and 'container(s)' in the abstract, keywords, or title.

```
(TITLE-ABS-KEY("MicroVM*") OR
TITLE-ABS-KEY("Unikernel*")) AND
TITLE-ABS-KEY("Container*") AND (LIMIT-TO(PUBYEAR,
2023) OR LIMIT-TO(PUBYEAR, 2022) OR LIMIT-TO(PUBYEAR,
2021) OR LIMIT-TO(PUBYEAR, 2020) OR LIMIT-TO(PUBYEAR,
2019) OR LIMIT-TO(PUBYEAR, 2018) OR LIMIT-TO(PUBYEAR,
2017)) AND LIMIT-TO(LANGUAGE, "English")
```

Listing 1: Search query

This resulted in 59 results that were then sorted by citations. The first ten papers that fit the scope were chosen as the base literature. Those consist of articles that compare and benchmark containers and microVMs/unikernels [12, 28, 38, 41, 30, 39] and articles that propose new microVMs/unikernels technologies [2, 27, 7]. As the proposal of new container technologies by the latter group might lead to a potential conflict of interest when these proposed technologies where benchmarked by the authors against existing container solutions, extra care has been taken to examine these papers for possible biases.

2 Comparison

When comparing the different systems it is essential to consider various configurations. Please note that in certain sections, some combinations may be absent, because no studies have been done regarding this specific configuration. There are three different configurations that have been taken into account. A fully-fledged operating system has not been included in the comparison, as its overhead increases the memory footprint and the startup times to a point where that is not a viable option [27].

1. Docker
2. MicroVM running with a slim OS
3. MicroVM running a unikernel

2.1 Performance

2.1.1 Resource usage

In the following paragraph, the resource usage, as well as the hardware utilization is discussed for Docker, unikernels, and microVMs.

Memory usage: To analyze the memory usage, Marco et al. ran 1000 Micropython guests on one host system. While unikernels and Docker containers had a memory footprint of around 5GB, a minimalist operating system created with TinyX consumed 27GB, while a standard Debian virtual machine required a hefty 114GB of memory [27]. This is because when running a minimalist OS within a microVM, multiple instances of the kernel are running. In addition to that the memory overhead of a microVm also has to be taken into account which is about 3MB per VM with out a guest operating system in case of Firecracker [1]. While the authors suggest that the 22GB difference isn't significant due to current memory prices, it's worth highlighting that the five-times increase in the memory footprint of the tiniyx VM instance compared to containers is considerable. For example, a Google VM with 8 cores and 8 GB Ram is 145.45$ per month, while the same machine with 32 GB Ram is 196.67$ [13].
According to Plauth et al., when running a simple HTTP Server, a Docker container has a memory footprint of around 80MB, the unikernel Rumprun has one of about 190MB, roughly twice the size, while a standard Ubuntu VM features a memory footprint of 250MB [30]. Similar memory usage results are also reported by Li et al. [23] and Mavridis et al. who claim that unikernels take around two times more memory than Containers [28]. Differing from these results, Kuenzer et al. showed that unikernels can outperform containers, claiming that the minimal amount of memory needed to run an nginx on an Unikraft unikernel is 5MB, while a Docker container needs 7MB [20] (see figure 1).

CPU usage: With respect to the CPU utilization, Marco et al. state that Docker containers and unikernels exhibit the lowest resource utilization, with TinyX, running in a microVM peaking at approximately 1% for 1,000 guests,

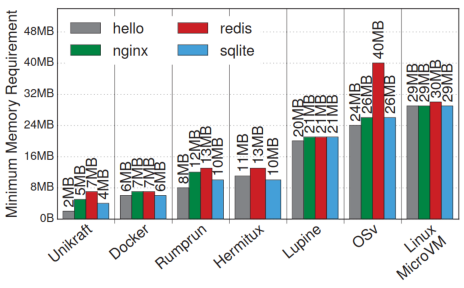


Figure 1: Minimal memory usage [20]

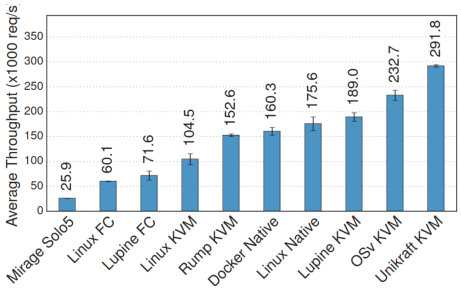


Figure 2: Requests per second [20]

while Debian VMs show poorer scalability, reaching around 25% utilization for the same number of VMs, which stems from the numerous services Debian runs by default [27]. Anjali et al. performed CPU benchmarks that showed that the CPU speed for unikernels and containers is roughly the same [2]. This is congruent with the findings of Plauth et al. [30].

Multiprocessing support is not widely spread around unikernels. For example, OSV does not support multiprocessing[11] and Unikraft only supports it since 2022 [37]. Therefore most of the benchmarks which have been examined here are single-core benchmarks.

The system call times are roughly similar when comparing a virtual machine deployed with KVM to the host. However, docker exhibits slightly slower performance, and gVisor experiences approximately 1.5 times slower execution when running on KVM [23].

According to Kuenzer et al., across four applications Unikraft (run with KVM) used fewer resources than docker and achieved better performance [20] (see figure 2). When modifying the application to use the special low-level APIs offered by Unikraft the performance improvements are even bigger, and even outperform the same application running on bare-metal [20]. This is achieved by removing all overhead and unnecessary components using a dependency graph that helps determine which components are needed for an application to run.

The current research presents heterogeneous results when examining the performance of unikernels, microVMs, and containers. The consensus is that containers are quicker than microVMs running a lightweight OS. When comparing unikernels in microVMs with containers, they're similar in how well they perform. Modern unikernels, like Unikraft, can even outperform containers in some scenarios.

2.1.2 Startup time

When it comes to startup times unikernels in combination with a microVM seem to be faster than the startup time of a container. While Docker containers required 150ms to start up when no containers were running, the startup time

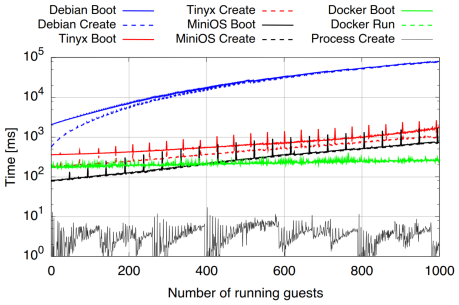


Figure 3: Startup time [27]

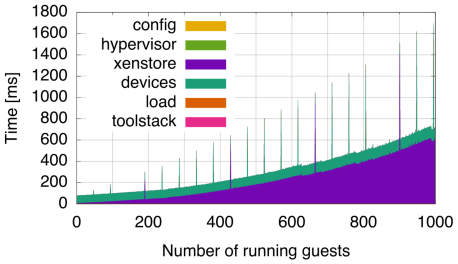


Figure 4: XenStore overhead [27]

increased to 1s when running 3000 Docker containers. On the other hand, unikernels booted within the range of 8-15ms (excluding the microVM), irrespective of the number of instances running. This time could even be reduced to 2.3ms when optimizing the unikernel for startup time [27]. However, these results were only achieved after improving upon some bottlenecks in the XenStore (which is a key-value store that allows the different components of the virtualization environment to share data [8]) which takes longer to update information the more VMs there are. However, it is possible to mitigate this problem not only by removing the XenStore [27] (see figure 4), but also by caching certain operations and thereby improving the performance for workloads with very many VMs >256 by 46% [8]. The final start to finish startup times can be compared in figure 3.

Goethals et al. found similar results when running Firecracker microVMs where they discovered that the Firecracker VM booted slightly faster than container-based alternatives, and OSv unikernels offer "extremely low boot times and significantly better performance than Docker containers" [12].

2.1.3 IO Performance

The IO performance is an important metric for modern cloud applications, as most communication happens over TCP/UDP [34]. Therefore a high IO throughput and low latency are essential for cloud infrastructure components. In the following, the unikernel OSv, Firecracker, the container runtime gVisor and runC are evaluated.

OSv significantly outpaces Docker on x86 architectures, exhibiting 50% faster response rates in completed requests per second. Conversely, Firecracker, using a lightweight OS, performs 20% worse. Further, containers deployed using gVisor achieve only 25% of the speed of those deployed with runC, demonstrating that the gVisor's emulated software kernel introduces substantial overhead for the benefit of enhanced security through improved isolation [12]. This is in line with Plauth et al. who found that the round-trip time for certain unikernels is smaller

than the round-trip time of docker, and almost matches native performance[30].

For ARM architectures, OSv unikernels offer the fastest processing, being 16% faster than Docker containers, while Firecracker running a lightweight OS only achieves 72% of the performance of OSv. Containers using gVisor only achieves 10% of the performance of containers running with runC.

When running using multiple processor cores, OSv and Firecracker scale worse than Docker and gVisor with more threads [12]. This is most likely because of missing or bad implementation of multi-core support [11].

According to Wang et al., Firecracker's I/O performance is behind Docker for small disk write sizes, but it catches up with (or exceeds) Docker when the records become larger [38]. Anjali et al.'s study confirms that Firecracker outperforms Docker by a factor of two for 4kb file sizes, which increases to a factor of four for 1MB file sizes [2].

It can be therefore concluded that virtual machines in combination with unikernels offer faster I/O, especially for larger chunks [38]. However, when it comes to handling HTTP requests through multiple cores, unikernels fall short due to their weak multi-core support, showing worse performance than containers [12].

2.2 Security

VMs, unikernels, and containers are widely used in the cloud to isolate multiple workloads. This has a lot of opportunities as this allows running multiple applications on a single host machine, but also has security challenges. In this section isolation and attack surface are compared to point out the trade-offs between the different technologies and to understand in which scenario which virtualization technology is most suitable. Isolation describes to which extent separation of workloads can be achieved, ensuring independent operation and no inference between those workloads. Attack surface refers to potential points of vulnerability that an attacker can exploit according to the definition of Ross et al. [32], while isolation is defined as the challenge of creating different virtual instances and keeping them independent [3].

Generally speaking, unikernels provide the highest level of isolation because each application is compiled with only the specific libraries and drivers it needs into a standalone kernel. As they do not rely on a host operating system, the risk of vulnerabilities that could be exploited from outside the unikernel can be reduced. VMs also offer a high degree of isolation, as each VM operates as a separate system with its own OS, libraries, binaries, and applications. This means that VMs are not affected by the operations of the host system or other VMs running on the same hardware. Containers provide the least amount of isolation because they share the host system's kernel [27] [38].

Compared to microVMs or containers, unikernels have technically the smallest attack surface because they do not have a native shell, unneeded code, and unnecessary system calls are removed or unsupported [36]. However, unikernels might overlook essential security features. A study from 2019 challenged the security advantages of two unikernels (Rumprun and IncludeOS), highlighting

the lack of important security features such as ASLR, W^X, Stack canaries, or heap integrity checks [29]. Rapp et al. examined a selection of unikernels based on selected security features, highlighting that the unikernel nano implemented the most features [22]. Another problems is that running the applications in kernel space gives an attacker who gained remote code execution direct access to kernel-level functionality where they have elevated access to system calls or raw packet I/O [36]. These can be pivot points for attacking neighboring hosts [29].

Containers on the other hand offer isolation with statically linked libraries and the ability to utilize security mechanisms at the host level [4]. They run within the user space, which means that all containers share the same kernel. In the event of a kernel compromise, all containers running on the host could also be compromised [24]. Those risks can be mitigated by following the guidelines of the official documentation and employing protection methods such as user-space isolation, host-based intrusion detection, rootless docker [31], or utilizing kernel security hardening mechanisms (e.g. Tripwire, SELinux) [39].

Virtualization relies on a hypervisor, which acts as an intermediary between the guest operating system and the underlying hardware. The level of isolation achieved is primarily attributed to the hypervisor's ability to leverage hardware features such as memory segmentation and hardware virtualization extensions. These extensions enable the hypervisor to operate in the host domain, while the guest virtual machines (VMs) function within the guest domain, enabling the guest VM kernels to run at ring level 0 [24]. It should be mentioned that unikernels usually do not run bare metal, which means that they also benefit from the security features of a hypervisor.

Williams et al. have explored yet another way to achieve the same level of isolation for unikernels similar to VMs by running unikernels as processes. By modifying the ukvm unikernel monitor, the monitor can load the unikernel into its own memory space. This modified monitor is called tender and switches to a restricted execution mode before executing the unikernel code. As a result, they were able to reduce the number of system calls by 50%, which in turn reduced the attack surface [40].

In conclusion, considering the security aspects of isolation and attack surface, the literature shows that microVMs offer strong isolation, while containers have limitations in that regard. Unikernels have strong isolation as well, but may lack privilege levels within the unikernel itself and fail to implement some of the security mitigations. To advance the security state of applications, a hybrid combination of these technologies holds potential [24]. During the review of several papers and unikernel projects, the evaluation of the unikernels security seems biased. The authors of Unikraft write that it should be possible to achieve good security while retaining high performance. However, important security features like ASLR and W$\hat{\text{X}}$ are not implemented [20]. Furthermore, through this analysis, a research gap has been identified, as none of the papers evaluated the security of unikernels running inside of microVMs, which could be a good combination as the microVM is addressing some of the before-mentioned missing security features of unikernels.

2.3 Usability and Tooling

In recent years, containers have emerged as the leading choice for deploying software applications and services among all container vendors. They have a robust ecosystem and a large community which has contributed to its popularity [16]. By building containers it can be made sure that the software runs on every cloud provider, effectively offering a way to mitigate vendor lock-in risks [17]. On the other hand, building unikernels is complicated as it requires a lot of custom configuration and build tools [27]. Projects like Unicraft aim to ease this process and reduce the complexity by providing build tools, libraries, and runners [20], however building docker images is still easier [9]. Unikernel's lack of a shell, online debugging, and potentially expensive source code recompilation for any updates or application changes [7]. This is a trade off they share with docker containers. This is also highlighted by Manco et al., stating that unikernels require significant development time due to the need for manual configurations, in addition to their lack of sophisticated tooling [27]. Because the tooling around container orchestration and building is superior compared to that of unikernels and microVMs [27] microVMs and unikernels have not penetrated the mainstream yet and are currently mostly used in AWS/Google functions where isolation is especially important, as the operations of one lambda function should not impact the operations of another lambda function.

The literature suggests that the tooling around microVMs is not that advanced as no major container orchestration system like Kubernetes supports the scheduling of microVMs out of the box [28]. However projects like kubevirt [19] extend Kubernetes using the Kubernetes CustomResourceDefinitions API [18], which allows the orchestration of resources that are not covered by the standard Kubernetes API [28] like virtual machines. This however has the drawback that according to Mavridis et al. running the VMs consumes two times more memory than running the containers [28] in some part because of the overhead of kubevirt.

To sum it up, unikernels are hard to build and maintain. Most unikernels are not at a place where every application can be run without problems and the debugging experience within a unikernel is inferior to the debugging experience inside a container. Furthermore, orchestration systems currently do not naively support microVMs and one has to rely on third-party solutions to get microVMs running within a Kubernetes cluster.

3 Results

While microVMs, in combination with a unikernel or a minimalist OS, continue to gain in performance, as of now, containers are still ahead. However, when combined with unikernels, microVMs demonstrate superior performance in some scenarios compared to containers. Furthermore, microVMs offer better isolation than containers because the microVMs do not share the host kernels, however unikernels lag in terms of security due to missing security features. It's however

worth mentioning that there are strategies to enhance the security of Docker containers.

One of the challenges with unikernels is their complex build process, and lack of robust development tools, community support, and orchestration tools, but these conditions are slowly improving, especially with the integration of microVMs running unikernels with Kubernetes using third-party libraries.

To answer the question of whether and where technologies like microVMs and unikernels could potentially replace containers the following conclusion can be drawn: Using microVMs with unikernels is a great fit for large-scale projects that can afford to take more time in development, especially when they need high performance and isolation. Another reason to use unikernels in combination with microVMs is to benefit from the fast boot times that unikernels offer in comparison to containers and slim Linux Operating systems. However, the high hurdle of building unikernels and orchestrating them within Kubernetes makes them only a viable option if there are not enough resources available for docker containers (for example when running on the edge), or if the very fast startup times in combination with high isolation needs are necessary. MicroVMs in combination with a small Linux kernel however are a good fit if only a higher isolation is needed. This is the case for applications like AWS Lambda where using microVMs allows AWS to run multiple different lambda functions on the same machine without endangering the integrity and security of other lambda functions. By being able to run different lambdas on the same machine the hardware utilization can be increased.

4 Discussion

The field of unikernel and microVM research is a relatively new one and in active development. Especially the tooling for unikernel has to improve if unikernels ever want to become a viable alternative to containers.

One notable research gap in the field of virtualization and containerization is the performance comparison between a Kubernetes (k8s) cluster running containers and a k8s cluster utilizing MicroVMs and unikernels. While there has been done some work by Mavridis and Karatza in [28] it is not as extensive as one would hope and still leaves some room for new investigations.

Despite numerous claims that unikernels are challenging to operate, which leads to increased development time [9, 7, 27], no comparative study has been conducted. It would be valuable to have a study similar to the one by Gleison et al., which involved a controlled experiment with 22 students who were assigned to implement eight queries in GraphQL and REST for accessing a web service to empirically assess the effort necessary to implement queries in these languages [5].

References

[1] Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation: February 25-27, 2020, Santa Clara, CA, USA. USENIX Association, Berkeley, CA (2020)

[2] Anjali, Caraza-Harter, T., Swift, M.M.: Blending containers and virtual machines. In: Nagarakatte, S. (ed.) Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. pp. 101–113. ACM Digital Library, Association for Computing Machinery, New York,NY,United States (2020). https://doi.org/10.1145/3381052.3381315

[3] van de Belt, J., Ahmadi, H., Doyle, L.E.: Defining and surveying wireless link virtualization and wireless network virtualization. IEEE Communications Surveys & Tutorials **19**(3), 1603–1627 (2017)

[4] Bias, R.: Unikernels will create more security problems than they solve (Dec 2021), `https://thenewstack.io/unikernels-will-create-security-problems-solve/`

[5] Brito, G., Valente, M.T.: REST vs graphql: A controlled experiment. CoRR **abs/2003.04761** (2020), `https://arxiv.org/abs/2003.04761`

[6] Bui, T.: Analysis of docker security. https://doi.org/10.48550/arXiv.1501.02967

[7] Chen, S., Zhou, M.: Evolving container to unikernel for edge computing and applications in process industry. Processes **9**(2), 351 (2021). https://doi.org/10.3390/pr9020351

[8] Chiba, D.J.: Optimizing Boot Times and Enhancing Binary Compatibility for Unikernels. Ph.D. thesis, Virginia Tech (2018)

[9] Cross, J.R.: Analysis of unikernels for load balancing and backend service deployment (2021)

[10] Elsevier: Scopus, `https://www.scopus.com`

[11] Goethals, T., Sebrechts, M., Al-Naday, M., Volckaert, B., de Turck, F.: A functional and performance benchmark of lightweight virtualization platforms for edge computing (2022). https://doi.org/10.1109/EDGE55608.2022.00020

[12] Goethals, T., Sebrechts, M., Atrey, A., Volckaert, B., De Turck, F.: Unikernels vs containers: An in-depth benchmarking study in the context of microservice applications. In: 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2). pp. 1–8 (2018). https://doi.org/10.1109/SC2.2018.00008

[13] Google: Google cloud platform (2023), https://console.cloud.google.com/compute

[14] Grattafiori, A.: Understanding and hardening linux containers. Whitepaper, NCC Group (2016)

[15] Jian, Z., Chen, L.: A defense method against docker escape attack. In: Proceedings of the 2017 International Conference on Cryptography, Security and Privacy. p. 142–146. ICCSP '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3058060.3058085, https://doi.org/10.1145/3058060.3058085

[16] Kithulwatta, W.M.C.J.T., Wickramaarachchi, W.U., Jayasena, K.P.N., Kumara, B.T.G.S., Rathnayaka, R.M.K.T.: Adoption of docker containers as an infrastructure for deploying software applications: A review. In: Saeed, F., Al-Hadhrami, T., Mohammed, E., Al-Sarem, M. (eds.) Advances on Smart and Soft Computing. pp. 247–259. Springer Singapore, Singapore (2022)

[17] Kratzke, N., et al.: Lightweight virtualization cluster how to overcome cloud vendor lock-in. Journal of Computer and Communications (12), 1 (2014)

[18] Kubernetes: (Mar 2023), https://kubernetes.io/docs/tasks/extend-kubernetes/custom-resources/custom-resource-definitions/

[19] Kubevirt, K.: Kubevirt/kubevirt: Kubernetes virtualization api and runtime in order to define and manage virtual machines. (May 2023), https://github.com/kubevirt/kubevirt

[20] Kuenzer, S., Badoiu, V.A., Lefeuvre, H., Santhanam, S., Jung, A., Gain, G., Soldani, C., Lupu, C., Teodorescu, Ş., Raducanu, C., et al.: Unikraft: fast, specialized unikernels the easy way. In: Barbalace, A. (ed.) Proceedings of the Sixteenth European Conference on Computer Systems. pp. 376–394 (2021)

[21] Kulkarni, A.V., Aziz, B., Shams, I., Busse, J.W.: Comparisons of citations in web of science, scopus, and google scholar for articles published in general medical journals. Jama (10), 1092–1096 (2009)

[22] Leonard Rapp, E.S.: Missing or weak mitigations in various unikernels (May 2022), https://x41-dsec.de/news/missing-or-weak-mitigations-in-various-unikernels/

[23] Li, G., Takahashi, K., Ichikawa, K., Iida, H., Thiengburanathum, P., Phannachitta, P.: Comparative performance study of lightweight hypervisors used in container environment pp. 215–223. https://doi.org/10.5220/0010440502150223

[24] de Lucia, M.J.: A survey on security isolation of virtualization, containers, and unikernels

[25] Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., Crowcroft, J.: Unikernels: Library operating systems for the cloud. ACM SIGARCH Computer Architecture News **41**(1), 461–472 (2013). https://doi.org/10.1145/2490301.2451167

[26] Madhavapeddy, A., Scott, D.J.: Unikernels: Rise of the virtual library operating system. Queue **11**(11), 30–44 (2013). https://doi.org/10.1145/2557963.2566628

[27] Manco, F., Lupu, C., Schmidt, F., Mendes, J., Kuenzer, S., Sati, S., Yasukata, K., Raiciu, C., Huici, F.: My vm is lighter (and safer) than your container. In: Association for Computing Machinery-Digital Library, ACM Special Interest Group on Operating Systems (eds.) Proceedings of the 26th Symposium on Operating Systems Principles. pp. 218–233 (2017)

[28] Mavridis, I., Karatza, H.: Orchestrated sandboxed containers, unikernels, and virtual machines for isolation-enhanced multitenant workloads and serverless computing in cloud. Concurrency and Computation: Practice and Experience **35**(11), e6365 (2023)

[29] Michaels, S., Dileo, J.: Assessing unikernel security. Technical report, NCC group, Tech. Rep. (2019)

[30] Plauth, M., Feinbube, L., Polze, A.: A performance survey of lightweight virtualization techniques. In: Service-Oriented and Cloud Computing: 6th IFIP WG 2.14 European Conference, ESOCC 2017, Oslo, Norway, September 27-29, 2017, Proceedings 6. pp. 34–48 (2017)

[31] Rahmansyah, R., Suryani, V., Yulianto, F.A., Ab Rahman, N.H.: Reducing docker daemon attack surface using rootless mode. In: 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM). pp. 499–502. IEEE (2021)

[32] Ross, R., Pillitteri, V., Dempsey, K., Riddle, M., Guissanie, G.: Protecting controlled unclassified information in nonfederal systems and organizations. Tech. rep., National Institute of Standards and Technology (2019)

[33] Sarkale, V.V., Rad, P., Lee, W.: Secure cloud container: Runtime behavior monitoring using most privileged container (mpc). In: ieee (ed.) 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud 2017). pp. 351–356. IEEE, Piscataway, NJ (2017). https://doi.org/10.1109/CSCloud.2017.68

[34] Sharp, H., Kolkman, O.: Discussion paper: An analysis of the 'new ip'proposal to the itu-t. Internet Society (2020)

[35] Snyder, H.: Literature review as a research methodology: An overview and guidelines. Journal of Business Research pp. 333–339 (2019). https://doi.org/https://doi.org/10.1016/j.jbusres.2019.07.039

[36] Talbot, J., Pikula, P., Sweetmore, C., Rowe, S., Hindy, H., Tachtatzis, C., Atkinson, R., Bellekens, X.: A security perspective on unikernels. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). pp. 1–7 (2020)

[37] Unikraft: (Jun 2022), https://unikraft.org/blog/2022-06-13-unikraft-releases-hyperion/

[38] Wang, Z.: Can "micro vm" become the next generation computing platform?: Performance comparison between light weight virtual machine, container, and traditional virtual machine pp. 29–34 (2021). https://doi.org/10.1109/CSAIEE54046.2021.9543457

[39] Watada, J., Roy, A., Kadikar, R., Pham, H., Xu, B.: Emerging trends, techniques and open issues of containerization: a review. IEEE Access **7**, 152443–152472 (2019)

[40] Williams, D., Koller, R., Lucina, M., Prakash, N.: Unikernels as processes. In: Computing, S..A.S.o.C. (ed.) Proceedings of the ACM Symposium on Cloud Computing. pp. 199–211. ACM Conferences, Association for Computing Machinery, New York,NY,United States (2018). https://doi.org/10.1145/3267809.3267845

[41] Xavier, B., Ferreto, T., Jersak, L.: Time provisioning evaluation of kvm, docker and unikernels in a cloud platform. In: ieee (ed.) 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). pp. 277–280. IEEE (2016). https://doi.org/10.1109/CCGrid.2016.86, https://repositorio.pucrs.br/dspace/bitstream/10923/14178/2/Time_provisioning_evaluation_of_KVM_Docker_and_Unikernels_in_a_Cloud_Platform.pdf