Course Web Service and Cloud Systems 2019-2020

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[3], and give 15 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. To introduce the students to scientific paper analysis, 4 scientific papers are analysed and discussed during the lecture hours. Reports are checked for plagiarism using Trinity tool integrated in Canvas (similarity score tolerated is max 20%).

[1] https://studiegids.uva.nl/xmlpages/page/2020-2021/zoek-vak/vak/79525
[2] https://masters.vu.nl/en/programmes/computer-science-big-data-engineering/index.aspx
[3] Formatting requirements: NeurIPS 2019 conference. More information and LaTeX templates can be found here: https://nips.cc/Conferences/2019/PaperInformation/StyleFiles

Fog Computing And Relation to Cloud Computing - A Review on Fog Computing And Privacy

Martijn, Lucien
lucienmartijn@gmail.com
12969168

Meijerink, Stijn
s.d.j.meijerink@student.vu.nl
12970247

Herold, Leonard
mail@lherold.me
12854743

Abstract

This literature study researches the state of Fog Computing with a focus on the privacy aspects. The heterogeneous, geo-distributed, virtualized computing paradigm is frequently used for real-time applications that require low-latency. This work identifies seven privacy concepts based on an established research methodology, each investigated in-depth in order to identify the current privacy issues and the corresponding state-of-the-art solutions in Fog Computing. Furthermore, multiple aspects are proposed in which both Fog Computing and Cloud Computing can benefit in order to achieve improved privacy. As a major finding, it turned out that there is a significant gap in the existing literature concerning the legal perspective of Fog Computing. Finally, this work can be used as a starting point to conduct an exhaustive study on the synergies between the Cloud and Fog, following this work's approach.

1 Introduction

The Internet of Things (IoT), according to a PWC study [11], has seen constant growth over the last decade, and the number of IoT devices deployed in fields such as industry, healthcare, and the consumer market grew steadily. For example, Gartner forecasts that 5.8 billion IoT devices will be installed in 2020, a 21% increase from 2019 [22]. Naturally, the massive growth in device numbers increases the amount of data generated. As IoT devices often have limited resources available (e.g., computational power), work is offloaded to the Cloud. However, IoT's real-time processing requirements and the long distance between the edge and the core of the network where the Cloud is located causes issues such as high latency and high bandwidth usage, making them both an unsatisfactory fit.

Fog Computing (FC), a term coined by Cisco in 2012, is a highly virtualized distributed computing paradigm that extends the reach of the Cloud to the edge of the network. It characterizes low latency, heterogeneity, mobility, location awareness, and geo-distribution [8]. Since the emergence of FC, the paradigm has continuously matured, and applications in different fields such as healthcare, smart homes, and vehicular ad hoc networks have been proposed [20, 23, 39]. In 2015 the OpenFog Consortium was founded as a global initiative backed by tech companies and academic institutions to standardize and stimulate the development of FC.

However, as it is an extension of the Cloud, it partly inherits issues present in Cloud Computing (CC), such as privacy and security. These issues are also relevant for FC. While solutions developed for CC can be transferred to the new paradigm, FC also introduces new privacy and security challenges. These issues are caused by the different inherent characteristics of FC, such as mobility, heterogeneity of Fog Nodes (FNs), and IoT devices. Consequently, the topic of data privacy has been hotly debated

in the last two decades. These discussions and the additional issues unique to FC motivate this work. Hence, this literature study aims to take a closer look at the current privacy issues in FC, answering three research questions:

1. What are current privacy concerns in the context of Fog Computing?
2. What are the current state-of-the-art solutions that address the privacy issues in FC and their respective advantages and drawbacks?
3. Can Cloud Computing adapt advancements made in Fog Computing or vice versa?

For the purpose of answering the research questions posed, this literature review's structure is as follows. First, the literature review methodology is presented in Section 2. The background and related work into FC are provided in Section 3. Section 4 aims to answer the questions posed in a structured matter, shedding light on the issues, the current state-of-the-art techniques, and solutions that can be transferred from or to CC. Finally, this work is concluded in Section 5 discussing the work, elaborating the limitations, and pointing out future challenges in FC.

2 Literature Review Methodology

This section regards the method used to investigate the scientific space concerning the research questions proposed in Section 1. First, the literature review methodology is presented thoroughly, including data sources and tooling. This not just aids independent reproducibility but also provides an understanding of how results were obtained. Next, the results of the literature query and selection process are presented.

To provide a comprehensive and objective view on the field and to comprehensively answer the research questions, a systematic literature review search process inspired by Brocke [9] was deemed beneficial for our research. Brocke divides the process into five steps: definition of review scope, the conceptualization of the topic, literature search, literature analysis, synthesis, and research agenda. The first three steps comprise this section, while the others are represented in the remainder of this work.

Accordingly, we focused our review on the research outcomes and application of FC with the goal of synthesizing the central issues in the field. The analysis is conducted in a neutral manner, which is aimed at general and specialized scholars (i.e., students and/or academic staff). Concerning literature selection, the analysis focused on representative and central works.

At first, the research context was conceptualized before the literature search was carried out, as suggested by Webster and Watson [40]. For this, the Scopus database was queried to retrieve surveys and reviews in the field of FC. Consulting the fields' top highly cited papers [29, 30, 42, 43] of the last five years provided better understanding, which allowed the formulation of the three research questions presented in Section 1.

Based on this, the initial Scopus search query returned 538 results without any refinement. Starting from the search keywords "Fog", "Computing", and "Privacy", we further narrowed the 538 results with Scopus' refine results options. We chose for all papers in the years 2012 up to 2020 because the term Fog computing was introduced in the paper of Bonomi et al. [8]. We further narrowed down by taking only the subject area Computer science; our main argument here is that the audience of this literature review is the class of Web Services and Cloud computing class. The publication stage is set to final, and source type is filtered to include Journal and Conference Proceeding only; hence the results comprise only peer-reviewed papers. After this process, the Scopus results include 402 results (see search query in Appendix A.1).

However, the final number of results returned by the search query was too high for a thorough evaluation. The results also contained papers that did not appear to be relevant to the research of this work. Thus, a step-by-step process was employed to reduce the number of results further to only relevant papers. These process steps, displayed in Table 1, included scans from high-level to detailed assessment. Each step was carried out using columns to indicate relevance in a Google Sheet. As a final result, 15 relevant papers were found, that during the review process were also assessed by backward and forward searches when applicable.

Table 1: Reduction Steps For Literature Search Results

Filter Action	Amount of Papers
Initial results	402
Remove duplicates	400
Remove wrong types (e.g., Conference Reviews)	368
Title Scan	262
Abstract Scan	62
Paper Scan	15

3 Background And Related Work

During our literature search, it became evident that the field shows ambiguous usage of terms (e.g., using FC and Edge Computing (EC) interchangeably). Therefore, this section establishes the relevant foundational knowledge required for the subsequent analysis and synthesis of the literature assessed by defining and delineating the required terminology. Furthermore, light is shed on other surveys that covered privacy-related aspects, and this work's contribution is highlighted.

3.1 The Computing Continuum - The Relation Between Edge, Fog And Cloud Computing

The computing paradigm landscape has witnessed big changes in the last decade, where data and compute at the edge of the network by edge resources have increased significantly [7]. There has been an increasing need to support real-time processing close to the data sources. However, the edge resources are still relatively limited in computing power and rely on the Cloud for more complex/heavy processing. Offloading of computation to the Cloud is characterized by shipping large amounts of data towards the core of the network, which is restrictive for the latency. Ultimately, this restriction affects the performance of the real-time requirement of applications at the edge of the network. Naturally, computational resources moved closer to the location, the edge, where it was needed [32]. With the emergence of Fog Computing (FC) as a new computing paradigm between the edge and the Cloud, the notion of a Computing Continuum is created. The aim of the Computing Continuum is to support data-driven applications by realizing an ecosystem where services and resources are aggregated in an on-demand manner. This work follows the clear distinction between EC and FC used by different authors [cf. 31, 32]. Edge Computing (EC) is characterized as a low resource computing paradigm. EC happens close to the end-user and comprises edge devices (i.e., smartphones or cars), that connect sensors to the higher level levels.

FC, therefore, is proposed as an extension to CC. By extending the Cloud to the edge, traffic to the Cloud is reduced, decreasing the latency and improving the Quality-Of-Service of IoT devices overall. Various concepts of applying FFC to IoT applications used in fields such as healthcare, smart homes, Software Defined Networking (SDN) and vehicular ad hoc networks have been proposed [20, 23, 39]. FC, as a concept, can be applied to a large range of fields and has a large potential. The need for standardization was the next logical step in order to improve the paradigm's security, scalability, and programmability. In 2015 the OpenFog Consortium was founded by tech companies and academic institutions from all over the world in order to standardize and stimulate the development of FC. Since 2019, the OpenFog consortium is merged with the Industrial Internet Consortium [25]. Although FC resolves the latency problem of the Cloud, it still relies on CC for computational and storage offloading (e.g., batch-processing) [32]. Consequently, FC is a connecting technology bridging EC and CC with each other, providing low-latency, bandwidth-efficient communication, higher scalability, and location awareness [2, 19]. The intermediary role of FC is further visualized in Figure 1, which gives more insight into the device and networking structures of the Computing Continuum.

3.2 Privacy And Security

Similar to the delineation needed between EC and FC, literature shows that distinguishing between privacy and security is challenging. In fact, security and privacy are closely related and are often discussed in parallel. However, as this work focuses on the privacy aspect of FC, a clear delineation is required.

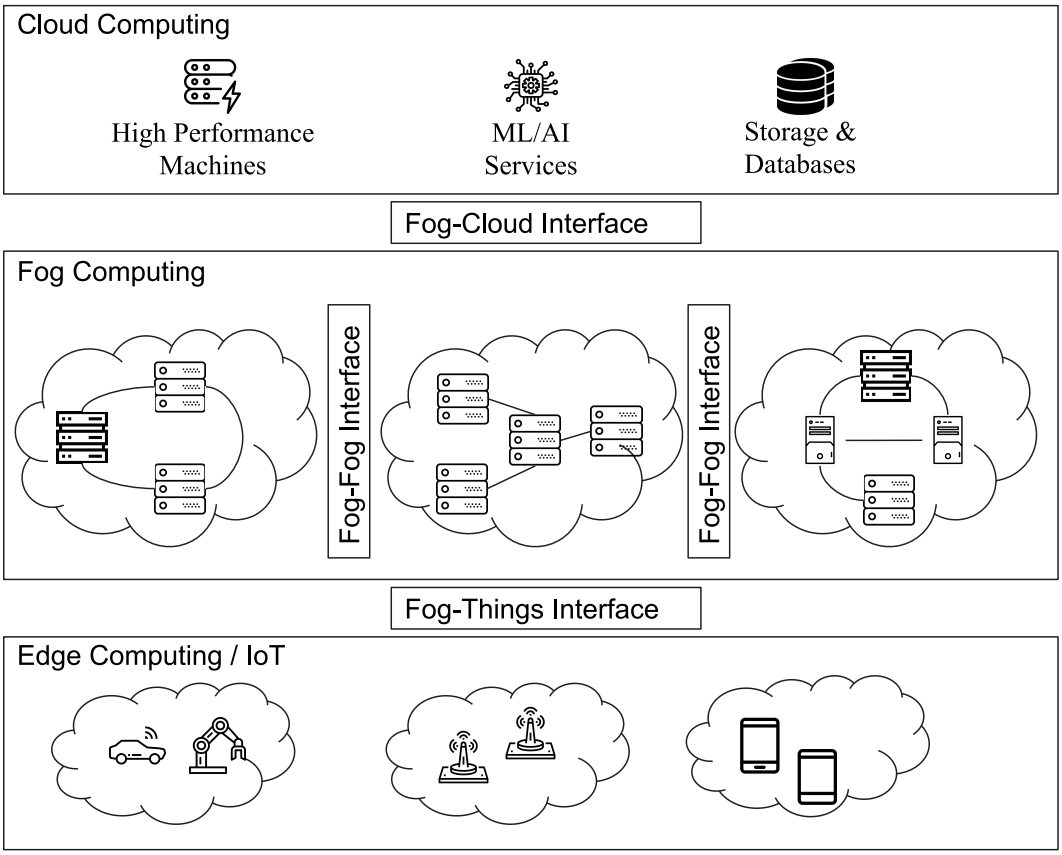


Figure 1: Computing Continuum - Fog Computing is bringing the edge and Cloud together, based on [7, 8, 31, 32]

.

Stojmenovic et al. [37] discuss security and privacy as two separate concepts. The authors discuss security on the basis of multiple malicious attacks, whereas they argue privacy is identified as the notion of hiding details. Rauf et al. [33] present a clear overview of security vs. privacy issues. Where security is always denoted by an evil actor trying to compromise a system, privacy is about which data is collected by a data processor, and how the data is handled in accordance with user consent. Other works that were consulted during the review do not distinguish between the two concepts clearly [cf. 4].

In this literature review, security is defined as the act of defending against attackers that employ attacks such as attack vectors or man-in-the-middle attacks, whereas other related papers describe privacy issues with a strong focus on security, mentioned in combination with several attacks. In contrast to that, this paper's notion of privacy focuses on the aspects that are centered around handling the data itself in a secure and reasonable manner.

3.3 Related Work

In this section, related reviews regarding privacy concerns in the field of FC are presented. The most influential papers were selected with a Scopus search on "Fog Computing Privacy" with the extra option to only show literature reviews.

Mao et al. [26] present a survey on Mobile Edge Computing (MEC) regarding different aspects. The authors discuss the challenges (among which privacy is one). The survey mostly focuses on MEC; however, the authors mention that FC and MEC do have an overlap, and the terminologies are frequently used interchangeably, as discussed previously.

Abbas et al. [1] start their paper 'Mobile Edge Computing a broad range of topics' by defining the different computing paradigms and the differences. Several applications are shown in which MEC can be used and continue with describing the benefits of MEC and how the state-of-the-art related research focuses on these topics. Most importantly, for this work, the authors touch upon the security and privacy issues of the computing paradigm, before concluding with the main benefits of MEC.

Hu et al. [21] discuss multiple facets of FC from the architecture to its open issues. First, the case is being made why FC is preferred above other edge-computing paradigms for real-time or latency-sensitive applications, and when network bandwidth is a bottleneck. The authors also make the comparison to CC and EC. Furthermore, the authors discuss the key technologies of FC and its applications. Relevant for this work is the section about security and privacy issues that are still ongoing, in which the authors discuss different concepts such as attacks, authentication, and access control.

Fernández-Caramés and Fraga-Lamas [15] wrote a review on Blockchain-based IoT in combination with FC. First, blockchains are introduced to the novice reader, and applications are discussed. Also, challenges and solutions with regard to privacy in Blockchain-based IoT are discussed; the authors specifically discuss identity certification and access management.

The contributions of this literature review lie in focus on privacy aspects in FC, covering the aspects more in-depth than previous works. As presented in the earlier paragraphs, the most influential papers discuss multiple aspects of EC, in which FC is one of the paradigms in the field. Further, these papers take a broader stance on privacy, e.g., incorporating low-level security aspects, and only briefly touch on the specifics on privacy.

This literature study, in contrast to the other related reviews, aims at specifically taking the privacy aspects into account and addressing security only if it is essential for understanding the privacy aspect.

4 Analysis - Privacy in Fog Computing

This section comprises the analysis regarding privacy in FC and is divided into three subsections elaborating on the specific research questions specified in Section 1. First, the privacy issues applicable to FC are presented and discussed. Next, the solutions and approaches found in the literature are described, while highlighting advantages and disadvantages. Last, it is discussed to which extent FC can apply approaches from CC, or if FC advances can be adapted.

4.1 Privacy Issues

As previously highlighted, privacy and security are concepts that are intertwined to some extent. Similarly, different issues that were identified in the literature needed a structured approach for analysis. Therefore, the assessment of FC's issues is presented based on the concept matrix (cf., Table 2), following Webster and Watson's recommendation for literature analysis [40]. Consequently, each area of concern regarding privacy is discussed alongside the concept matrix.

The concept of **Identity Management** in Table 2 refers to all aspects relevant for distinguishing a user. Furthermore, it also incorporates access control and trust. The issue of identity management in FC is of great importance because of multiple aspects. First of all, the leakage of the identity of the IoT device itself or the end-user can cause problems; this can happen when communicating with a FN. Most of the time, identity hiding happens through proxies; however, in FC, this makes the latency problem surface again [2, 24]. Another issue is that many IoT devices are not powerful enough to run cryptographic algorithms needed for authentication protocols [4, 31]. However, this is dependent on the application and devices used, as authors [cf. 2, 6, 21, 24] use encryption in combination with authorization. Other authors pose the problem of trust, where there are no efficient mechanisms to measure when and how to trust IoT devices [4]. Furthermore, the traditional trust models for CC can not directly be applied to FC because of the lack of centralized management [31]. Moreover, FC has vast amounts of shared data that need a fine-grained access control scheme. This is much harder than in CC [19]. Lastly, there are issues surrounding the distributed nature of FC. One example is that frequent re-authentication is essential in applications such as vehicular networks [16, 19].

Location Privacy is defined as the concept surrounding the leakage of device's and/or user's location data. The location itself consists of information based on the spatial correlation between FNs [cf. 2] and their connected devices or GPS/geospatial data. While in CC, an endpoint (i.e., a device) requesting a location-based service must explicitly send the location, FC can infer the location due to FN's location awareness [19]. Guan et al. [19] argue that FC does improve the location privacy, while other authors [2, 6, 24] claim that the issue of leakage of location information in FC is still imminent. In fact, it must be differentiated between the explicit sending of location information and the inferring of an approximate location. FC can provide location-based services without knowing the user's explicit geo-location due to the FN's location awareness; however, as a result, FC can suffer from indirect location privacy leakage because the end-devices' closeness to FNs can disclose location information of end-users [2, 24, 33].

The concept of **Data Confidentiality** is comprised of all measures to protect the data against unlawful access by others. A prime example of this is the data encryption in storage or during communication between different entities of a FC network [19, 24]. However, FC's low-latency objective and decentralized nature pose a challenge to pragmatic ways to secure data against such access (e.g., eavesdropping) in real-world scenarios such as vehicle to vehicle networks [32, 35]. Furthermore, given the trust issues, as already discussed previously, data stored on FN should be securely stored (encrypted) [19]. According to Hu et al. [21], such means should provide confidentiality, integrity, and availability. However, due to the low computational power of devices and sensors, such encryption seems to pose a challenge [6, 31]; still, Pape and Rannenberg [32] indicate that IoT devices and large edge devices (e.g., smartphones) become more powerful. Overall, it can be said that keeping data confidential throughout the entire FC space from the Cloud to the edge is a relevant issue for FC, requiring low computational cost cryptographic authentication and secure means of communication [31].

The concept of **Usage Behavior** centers around usage-patterns that, if accessible by third parties, can be used for inferring personal habits and sensitive patterns. For example, smart energy meters can be used to obtain sleeping habits based on power consumption [33]. In FC, this problem is exacerbated, as multiple entities, such as FNs, participate in providing the service to a user [6, 32]. From a technical perspective, usage patterns are mainly presented by data being send from end-devices to nodes in the Fog network [2]. As a result, a range of different usage habits can be disclosed; device movements, sleeping habits, or if a person is at home or not [1, 32, 33]. The authors that cover Usage Behavior argue that measures against the leakage of such information are necessary due to FC's inherent characteristics.

Table 2: Concept Matrix.

Paper	**Identity Management**	**Location Privacy**	**Data Confidentiality**	**Usage Behavior**	**Data Management**	**Malicious Attacks**	**Legal**
Abubaker et al. [2]	✗	✗		✗	✗		
Alrawais et al. [4]	✗						
Andrew & Karthikeyan [6]	✗	✗	✗	✗		✗	
Chertchom et al. [10]					✗		
Cui et al. [13]	✗				✗		
Ferrag et al. [16]	✗					✗	
Garg et al. [17]					✗		✗
Guan et al. [19]	✗	✗	✗		✗		
Hu et al. [21]	✗		✗		✗		
Li et al. [24]	✗	✗	✗		✗		
Mukherjee et al. [31]	✗		✗	✗		✗	
Pape and Rannenberg [32]			✗	✗	✗		
Rauf et al. [33]		✗		✗			
Ren et al. [34]					✗		
Ren et al. [35]	✗		✗				

As Stojmenovic et al. discuss, privacy is hiding details. The concept of **Data Management** taps into this, as it is involved with splitting and filtering data (at the edge), storing data, and offloading data. Firstly, IoT devices that are located on the edge in a distributed manner gather vast amounts of data [10]. However, these resource-limited devices outsource their data and the computing thereof, to FNs, which can result in privacy leakage [2, 13]. This can be counteracted by using more powerful IoT devices; however, this is not always feasible [17]. Furthermore, the responsibility of storing data can be moved from the edge device to the Cloud. Nonetheless, this causes the data-owner to give up its possibly private data [19]. Auditable secure Cloud data storage techniques are created to get insight into what happens with the data, but these cannot be applied directly to FC [19, 24]. Lastly, when performing actions such as querying the data or submitting data to the higher layers of the architecture, end-user's private data in the form of private parameters or as the data itself for a computation have the risk of being exposed to others [32, 34]. However, these actions are needed frequently, as FC is an extension of the Cloud and thus needs the Cloud for its more resource-intensive computations [21].

Defending against **Malicious Attacks** in the context of privacy preservation range from techniques such as intrusion detection systems [16, 31] and decoy technology [31]. If such techniques are not applied in FC, then there is little resistance against all kinds of attacks, ranging from Man-in-the-Middle to Sybil attacks. Andrew and Karthikeyan [6] describe many attacks to attack fog systems, but these attacks are not in the scope of this paper. Essentially, the main problem is that FNs are frequently susceptible to attacks because FNs, Cloud, and end-users are not in the same trusted domains [16]. However, the stance of this paper is that FNs cannot be fully trusted. This is in the same way as Mukherjee et al. reason stating: "a compromised node is a legitimate part of the network" [31, p. 19296].

With developments such as the General Data Protection Regulation (GDPR) [14] and the General Data Protection of the Federal Trade Commission of the United States of America, the **Legal** aspect of privacy is impossible to neglect. This new legislation introduces concepts such as personal data, ownership over data, and roles such as data controller and data processor [17]. FNs are placed closer to the user for processing the data on their behalf. However, by doing this, the user/IoT device hands over its local ownership to a (trusted) third party. Under the new legislation, it raises issues such as data ownership and liability in the case of a data breach [17].

4.2 Approaches And Solutions

Based on the previously identified issues, the following paragraphs discuss possible approaches and solutions in FC. Consequently, each of the issues is discussed separately to provide sufficient details that answer the second research question.

As can be seen in Table 2, there are several papers that address the issue of **Identity Management**. Namely, Abubaker et al. [2] propose to pair each FN with a Trusted Third Party (TTP) node. The purpose is to decouple the data (usage information) connected to the identity of the user. The TTP node holds the identity information, and the FN only holds the public data, which is to be processed. However, the paper does not describe the scenario in the existence of a rogue TTP node. A rogue TTP node could find out the identity of the end-user. Cui et al. [13] introduce a permissioned blockchain-based IoT data management system, where each FN keeps a copy of the ledger and is paired with an IoT device. Smart contracts stored on the immutable Blockchain (BC) enforce the Access-Control List rules of the end-user's device, which works well to defend against unauthorized access by rogue IoT devices. Similarly, Li et al. [24] propose an efficient and privacy-preserving carpooling scheme that fully preserves the end-users privacy using a private blockchain-assisted vehicular FC. Here, the authors' scheme uses an anonymous authentication scheme in order to authenticate users and identify malicious users. The private BC is used in order to record carpooling records for auditability purposes. Hu et al. [21] propose a privacy preservation scheme for FC based face identification. The full scheme consists of a session key agreement scheme and data encryption scheme to preserve the end-user's privacy in the complete communication process from the end-user to the Cloud and back. With the ever-increasing likelihood that a mature quantum computer will be realized in the distant future, implementations based on Diffie-Hellman and elliptic curve cryptography will be insecure and a threat to the end-users privacy [27]. Improvements on this scheme need to be thought out in order to be made quantum computer resistant.

In literature, different approaches for preserving **Location Privacy** data are proposed. Li et al. [24] propose a blockchain-assisted vehicular FC for the field of carpooling businesses. Notably, the location of users is protected as location information is not exposed further than one FN in the network. By this, the authors argue that the contents of the carpooling relevant messages are protected from access by the Cloud servers or other entities. However, as carpooling routes are stored on a permissioned BC, it is possible for the entities participating in the BC to unravel certain travel patterns of end-users, resulting in weaker location-privacy. Similarly, Abubaker et al.'s [2] approach aims to hinder other FNs or Cloud instances from inferring the true location of end-devices. A TTP node results in FNs being unable to retrieve the correct end-device's location, as it is not aware of the exact identity of a specific end-device. Further, the authors proposed using so-called mixes, which refer to collecting incoming messages of an FN and sending them out as a batch. This way, no link between a FN's incoming and outgoing message can be made. Consequently, the end-devices location is, based on the amount of batched messages, harder to infer by someone monitoring the FN.

In terms of **Data Confidentiality** solutions, it is evident that the limited computing capabilities provided by end-user devices at the edge require tailored solutions [35, 31]. As such, authors have proposed offloading the encryption and authentication responsibility to more conventional powerful devices (i.e., smartphones) [32]. Pape and Rannenberg specifically discuss the personal data store privacy pattern, that focuses on storing data locally. Based on the computational capabilities of the IoT device, data is stored on the device or a nearby trusted device (e.g., the home server). If the data is sensitive, data can also be encrypted so that only trusted devices can access it, and thus access by the Cloud or FNs is prevented [32]. Such a privacy pattern, however, is significantly based on the use-case of FC. For example, if FC or CC is required to process the data, this is not applicable. A popular approach to secure data communication in FC is SDN [6] and allows flexible configuration of accessibility by different entities in the FC space.

In order to obfuscate the **Usage Behavior**, a range of solutions that aim to increase the challenge of interfering patterns exist. These range from data separation to adding dummy data. Pape and Rannenberg [32] present four solutions based on the privacy patterns they follow: data isolation at different entities (e.g., an FN), adding noise to communication/measurements, vertical clustering of FNs and aggregation of data. Isolating the data at different entities in FC is beneficial for privacy, as it prevents FC from misusing the data as a whole. However, such a solution can handle both computation and storage, if it is accompanied by a computational engine that can run in a distributed manner. Further, adding noise to communication is only feasible if its scenario can handle such noise, as Pape and Rannenberg [32] correctly deduce. However, such limitations reduce the applicability significantly, as applications like a billing system cannot handle such noise well [32]. Abubaker et al. [2] propose a similar method by adding dummy tasks when tasks are offloaded [2].

In terms of **Data Management** and data-sensitive filtering, Chertchom et al. [10] propose a data management portfolio, containing two different kinds of Fog architectures that handle sensitive data of the end-users. The filtering of data is based on a 7-rule policy, as not all data is privacy sensitive. In the first architecture, the data is filtered in the Fog Layer and sent either to a Private Cloud that contains the privacy-sensitive data or to a Public Cloud that stores the data used for processing the general data. In the second architecture, the filtering of privacy data is applied at the edge layer, as certain applications at the edge require real-time processing. This is done by offloading the filtered data to the fog layer, where each FN is either responsible for processing privacy or general data. Cui et al. [13] propose an IoT management platform based on a permissioned BC and smart contracts. The data management model provides decentralized access control and authorization, which are enforced by means of immutable smart contracts. The policies defined in the smart contracts define how the data is managed for requests coming from the IoT devices. As the aim of the paper is the privacy of data in IoT devices in the fog, a permissioned BC does not offer the optimal security. Namely, even though there is no central server, only permissioned nodes in the BC network can vote. This does not provide Byzantine Fault Tolerance, which implies the permissioned BC is nothing more than a time-stamped list of entries [18]. In the privacy-preservation scheme of Hu et al. [21], the integrity of data is based on the SHA-1 hashing algorithm. However, since February 2017, SHA-1 has been subject to a successful collision attack. In light of this, a more secure hashing algorithm has to be employed in order to preserve data integrity [36].

Resiliency against **Malicious Attacks** can be achieved by deploying Intrusion Detection Systems (IDS) at all levels of the three-tier computing continuum, as proposed by Mukherjee et al. [31]. It is argued that if an IDS is deployed at just two levels (e.g., Fog and Edge), then there is a possibility of

intrusion from malicious software from a vulnerable node to the rest of the system. Coordination of the different detection components which are distributed over each level of the system is needed and can be achieved with the deployment of a perimeter IDS [12]. It could be the case that an IDS is implemented in a faulty manner or that there are software bugs in the code. Given that an attacker knows this, there is always a possibility of a breach into the system. User behavior profiling is a beneficial technique to monitor the amount and duration of access to sensitive data. Stolfo et al. [38] proposed a new approach in order to protect user data in the Fog and Cloud using decoy technology. The user's data is profiled, and subsequently, abnormal behavior patterns are detected. If an attack is suspected, the decoy sends a large amount of garbage data to the attacker. With this technique, the user's real sensitive data in the Fog and Cloud is protected. However, profiling user behavior could be harmful to the privacy of the user. New data regarding user behavior is generated and stored, which gives rise to the possibility of exploitation of the user's private data.

Regarding **Legal**, Garg et al. [17] discuss many recommendations to abide by the General Data Protection Regulation (GDPR). They focus on GDPR because this is stricter than the US's regulation. First of all, they propose that organizations have to adopt a *privacy by design* approach when an IoT device collects personal identifiable information [17]. Moreover, a protection officer has to be appointed to handle the accountability aspect in case of a data breach [17]. Additionally, when fog devices are being used, storing data in a distributed manner, and thus having multiple copies of the data, the organization should ensure that these copies are deleted when user consent has been withdrawn, or when the purpose of the data has become useless [17]. Moreover, Pape and Rannenberg [32] describe a storage system to store data on the devices. In this way, the data owner stays in control of its data. This, however, is not applicable to every application. Garg et al. [17] continue with the notion that pseudonymization should be used when it is crucial to process sensitive or personal identifiable information. Additionally, the amount of metadata should be kept low, only so that the tasks needing to be performed can be performed, such as querying [17]. Lastly, in a multi-tenant situation, the storage and processing of data should be separated both logically and physically. However, a critical note should be made; Garg et al. [17] discuss all the legal recommendations based on standardized architectures and use cases, which makes these recommendations no guarantee.

4.3 Synergies Toward Improved Privacy

In the previous two subsections, issues and related solutions in the context of FC are presented. As FC can be argued to be a related paradigm to CC, this work also aims to evaluate to which extent FC and CC can profit from each other's solutions. Specifically, the following paragraphs assess whether the solutions found in the underlying literature search can be applied, or the shortcomings in FC can benefit from advancements in CC. Overall, two synergies could be materialized based on the synthesized literature. However, additional work, as discussed in Section 5, could yield more such synergies.

Given that CC has been criticized throughout the years [41] regarding privacy, it seems evident that FC can adopt certain aspects. A prominent case this literature study has identified are the legal issues of the Cloud that have been addressed by all major commercial players promoting Cloud certifications as a standard practice [see 3]. However, during the literature review, no evidence was found that certification is being applied to FC. The usage of such certifications could be beneficial in multiple aspects. For example, it could resolve trust issues if a user questions whether he or she can trust a fog provider to store and delete data according to the GDPR. Additionally, such certifications could certify that diagnostic data that is collected by FNs cannot be used to infer sensitive usage behavior.

Vice versa, Pape and Rannenberg's [32] presented privacy patterns for FC, which could be an approach worthwhile assessing for CC. Especially since CC is tightly coupled with FC. The privacy patterns deemed most promising during the review are Personal Data Store, Data Isolation at Different Entities, and Added Noise Measurement Obfuscation. Concerning the isolation of data at different entities, it could be argued that this can be seen as a part of the Hybrid Cloud model. Based on the pattern, this could be extended to a scenario where data is stored across Public Clouds using a standardized, openly accessible storage format. For example, unifying the protocol of Amazon's S3 [5] and Microsoft's Azure Storage [28]. Furthermore, the Personal Data Store pattern locks personal data from Cloud Native applications by keeping personal data on local devices only. This could alleviate concerns from businesses that currently refrain from using the Cloud due to these privacy issues (i.e., health care- and banking sector).

5 Discussion And Conclusion

The findings of this literature review are three-fold, answering three research questions based on a systematized literature review.

Based on our concept matrix, the findings show that most literature finds issues revolving around identity management, data confidentiality, and data management (cf. Section 4.1). As shown in Section 4.2, these issues can be addressed in most cases by the use of encryption strategies, authentication schemes using signatures, and data-sensitive filtering in Fog architectures where the data is split and delegated to separated data providers. When comparing FC and CC's privacy solutions, we identified some opportunities for improvement, such as certification. This is a regular practice in CC, but the literature study has not been able to identify that this is also the case for FC. Furthermore, it is argued that evaluating a possible application of the privacy patterns described by Pape and Rannenberg [32] could be beneficial for CC.

However, given the limitations that are elaborated upon below, the findings indicated are also partly related to security aspects. Therefore, specifically the issues of identity management, data confidentiality, and malicious attacks can also be seen as security-related, and as such, should be further discussed with a more in-depth security assessment. Nevertheless, as they do impact privacy as discussed in Section 4, they were kept to present the entire process of our analysis.

Furthermore, other works have closely assessed the field of privacy and security (e.g., [4, 16, 19]). The paper at hand contributes to the referenced papers by synthesizing the issues in sole regard to privacy, while the other authors have intertwined their assessments. This has also shown that existing literature have difficulty in distinguishing security from privacy and vice versa in their research findings. This work focused on privacy in FC; however, limitations presented below are present and can be aided by further research.

The limitations of this literature study are reasoned in the limited resources and the narrowly defined scope of this work. Several activities required thorough preparation and were time-intensive. Specifically, defining a well-scoped research topic (i.e., privacy in FC), deciding on an established literature methodology, and the actual execution of the literature review. Therefore, this work focused only on the central and pivotal papers that were found. This poses the possibility that relevant literature was not captured during the review. However, an exhaustive literature search was beyond the scope of this work.

Nonetheless, this work has shown how an exhaustive literature review with the focus on privacy in FC can be carried out using a well-established methodology. It also shows that using Webster and Watson's [40] concept matrix to guide the reader can be helpful in structuring the evaluation.

To conclude, this literature study answered the three research questions. Overall, seven privacy-related concepts were identified based on the issues that were highlighted in Section 4.1. Furthermore, a selection of possible solutions was briefly described, covering possible advantages and drawbacks (cf. Section 4.2). These concepts were the input for the analysis in Section 4.3 to identify how the two computing paradigms CC and FC could improve on the basis of privacy. It was shown that solutions in CC exist and potentially be applied to FC and vice versa.

Consequently, this work indicates that an exhaustive and in-depth review of the privacy concepts in both paradigms is required, and an assessment of whether FC and CC can benefit from each other's solutions is aimed at improving upon the privacy issues present. However, the FC field suffers from a lack of a legal perspective, and thus a current gap of FC's privacy has been identified that centers around legal considerations and certification programs.

References

[1] Nasir Abbas et al. "Mobile Edge Computing: A Survey". In: *IEEE Internet of Things Journal* 5.1 (Feb. 2018), pp. 450–465. ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2750180.

[2] Nabil Abubaker, Leonard Dervishi, and Erman Ayday. "Privacy-preserving fog computing paradigm". In: *2017 IEEE Conference on Communications and Network Security (CNS)*. Las Vegas, NV: IEEE, Oct. 2017, pp. 502–509. ISBN: 9781538606834. DOI: 10.1109/CNS.2017.8228709.

[3] Cloud Security Alliance. *STAR | Cloud Security Alliance*. 2020. URL: https://cloudsecurityalliance.org/star/#_registry (visited on 05/27/2020).

[4] Arwa Alrawais et al. "Fog Computing for the Internet of Things: Security and Privacy Issues". In: *IEEE Internet Computing* 21.2 (Mar. 2017), pp. 34–42. ISSN: 1089-7801. DOI: 10.1109/MIC.2017.37.

[5] Amazon Inc. *Amazon S3*. 2020. URL: https://aws.amazon.com/s3/ (visited on 05/27/2020).

[6] J. Andrew and J. Karthikeyan. "Privacy-Preserving Internet of Things: Techniques and Applications". en. In: *International Journal of Engineering and Advanced Technology* 8.6 (Aug. 2019), pp. 3229–3234. ISSN: 2249-8958. DOI: 10.35940/ijeat.F8830.088619.

[7] L. Atzori, A. Iera, and G. Morabito. "The Internet of Things: A survey". In: *Computer Networks* 54.15 (2010), pp. 2787–2805. DOI: 10.1016/j.comnet.2010.05.010.

[8] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012, pp. 13–16. DOI: 10.1145/2342509.2342513.

[9] Jan vom Brocke et al. "Reconstructing the giant: On the importance of rigour in documenting the literature search process". In: (2009). URL: http://aisel.aisnet.org/ecis2009/372/.

[10] Prajak Chertchom et al. "Data Management Portfolio for Improvement of Privacy in Fog-to-cloud Computing Systems". In: *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)*. Toyama, Japan: IEEE, July 2019, pp. 884–889. ISBN: 9781728126272. DOI: 10.1109/IIAI-AAI.2019.00179.

[11] Raman Chitkara et al. *The Internet of Things: the next growth engine for the semiconductor industry*. 2015. URL: https://www.pwc.de/de/technologie-medien-und-telekommunikation/assets/pwc-studie-prognostiziert-boom-in-der-halbleiterbranche.pdf (visited on 05/27/2020).

[12] Tiago Cruz et al. "A cybersecurity detection framework for supervisory control and data acquisition systems". In: *IEEE Transactions on Industrial Informatics* 12.6 (2016), pp. 2236–2246. DOI: 10.1109/TII.2016.2599841.

[13] Hongyan Cui et al. "IoT Data Management and Lineage Traceability: A Blockchain-based Solution". In: *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*. Changchun, China: IEEE, Aug. 2019, pp. 239–244. ISBN: 9781728107387. DOI: 10.1109/ICCChinaW.2019.8849969.

[14] *EUR-Lex - 32018R1725 - EN - EUR-Lex*. May 2018. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018R1725 (visited on 05/23/2020).

[15] T. M. Fernández-Caramés and P. Fraga-Lamas. "A Review on the Use of Blockchain for the Internet of Things". In: *IEEE Access* 6 (2018), pp. 32979–33001. DOI: 10.1109/ACCESS.2018.2842685.

[16] Mohamed Amine Ferrag et al. "Privacy-preserving Schemes for Fog-based IoT Applications: Threat models, Solutions, and Challenges". In: *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*. El Oued: IEEE, Oct. 2018, pp. 37–42. ISBN: 9781538694930. DOI: 10.1109/SaCoNeT.2018.8585538.

[17] R. Garg, Sz. Varadi, and A. Kertesz. "Legal Considerations of IoT Applications in Fog and Cloud Environments". In: *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. Pavia, Italy: IEEE, Feb. 2019, pp. 193–198. ISBN: 9781728116440. DOI: 10.1109/EMPDP.2019.8671620.

[18] Vincent Gramoli. "On the danger of private blockchains". In: *Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL'16)*. 2016.

[19] Y. Guan et al. “Data Security and Privacy in Fog Computing”. In: *IEEE Network* 32.5 (2018), pp. 106–111. DOI: 10.1109/MNET.2018.1700250.

[20] Mohammad Aminul Hoque and Ragib Hasan. “Towards an Analysis of the Architecture, Security, and Privacy Issues in Vehicular Fog Computing”. In: May 2019. DOI: 10.1109/SoutheastCon42311.2019.9020476.

[21] Pengfei Hu et al. “Survey on fog computing: architecture, key technologies, applications and open issues”. en. In: *Journal of Network and Computer Applications* 98 (Nov. 2017), pp. 27–42. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2017.09.002.

[22] Gartner Inc. *Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020*. 2019. URL: https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io (visited on 05/23/2020).

[23] Jung-Hoon Lee, Sang-Hwa Chung, and Won-Suk Kim. “Fog server deployment technique: An approach based on computing resource usage”. In: *International Journal of Distributed Sensor Networks* 15 (Jan. 2019), p. 155014771882399. DOI: 10.1177/1550147718823994.

[24] M. Li, L. Zhu, and X. Lin. “Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing”. In: *IEEE Internet of Things Journal* 6.3 (2019), pp. 4573–4584. DOI: 10.1109/JIOT.2018.2868076.

[25] Electronics Mags. *The Industrial Internet Consortium and OpenFog unite to get momentum of business internet*. 2019. URL: https://electronicsmags.com/the-industrial-internet-consortium-and-openfog-unite-to-get-momentum-of-business-internet/ (visited on 05/28/2020).

[26] Yuyi Mao et al. “A Survey on Mobile Edge Computing: The Communication Perspective”. In: *IEEE Communications Surveys Tutorials* 19.4 (2017), pp. 2322–2358. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2745201.

[27] Vasileios Mavroeidis et al. “The impact of quantum computing on present cryptography”. In: *arXiv preprint arXiv:1804.00200* (2018). DOI: 10.14569/IJACSA.2018.090354.

[28] Microsoft Inc. *Azure Storage - Secure cloud storage | Microsoft Azure*. 2020. URL: https://azure.microsoft.com/en-us/services/storage/ (visited on 05/27/2020).

[29] Carla Mouradian et al. “A comprehensive survey on fog computing: State-of-the-art and research challenges”. In: *IEEE Communications Surveys & Tutorials* 20.1 (2017), pp. 416–464. DOI: 10.1109/COMST.2017.2771153.

[30] M. Mukherjee, Lei Shu, and Di Wang. “Survey of fog computing: Fundamental, network applications, and research challenges”. In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 1826–1857. DOI: 10.1109/COMST.2018.2814571.

[31] M. Mukherjee et al. “Security and Privacy in Fog Computing: Challenges”. In: *IEEE Access* 5 (2017), pp. 19293–19304. DOI: 10.1109/ACCESS.2017.2749422.

[32] S. Pape and K. Rannenberg. “Applying Privacy Patterns to the Internet of Things’ (IoT) Architecture”. In: *Mobile Networks and Applications* 24.3 (2019), pp. 925–933. DOI: 10.1007/s11036-018-1148-2.

[33] A. Rauf, R.A. Shaikh, and A. Shah. “Security and privacy for IoT and fog computing paradigm”. In: 2018, pp. 96–101. DOI: 10.1109/LT.2018.8368491.

[34] H. Ren et al. “Querying in Internet of Things with Privacy Preserving: Challenges, Solutions and Opportunities”. In: *IEEE Network* 32.6 (2018), pp. 144–151. DOI: 10.1109/MNET.2018.1700374.

[35] J. Ren et al. “A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet”. In: *ACM Computing Surveys* 52.6 (2019). DOI: 10.1145/3362031.

[36] Marc Stevens et al. “The first collision for full SHA-1”. In: *Annual International Cryptology Conference*. Springer. 2017, pp. 570–596. DOI: 10.1007/978-3-319-63688-7_19.

[37] I. Stojmenovic et al. “An overview of Fog computing and its security issues”. In: *Concurrency Computation* 28.10 (2016), pp. 2991–3005. DOI: 10.1002/cpe.3485.

[38] Salvatore Stolfo, Malek Salem, and Angelos Keromytis. “Fog Computing: Mitigating Insider Data Theft Attacks in the Cloud”. In: May 2012, pp. 125–128. ISBN: 978-1-4673-2157-0. DOI: 10.1109/SPW.2012.19.

[39] Wenjuan Tang et al. “Lightweight and Privacy-Preserving Fog-Assisted Information Sharing Scheme for Health Big Data”. In: Dec. 2017, pp. 1–6. DOI: 10.1109/GLOCOM.2017.8254989.

[40] Jane Webster and Richard T Watson. “Analyzing the past to prepare for the future: Writing a literature review”. In: *MIS quarterly* (2002), pp. xiii–xxiii. URL: https://www.jstor.org/stable/4132319.

[41] Zhifeng Xiao and Yang Xiao. “Security and privacy in cloud computing”. In: *IEEE communications surveys & tutorials* 15.2 (2012), pp. 843–859. DOI: 10.1109/SURV.2012.060912.00182.

[42] Ashkan Yousefpour et al. “All one needs to know about fog computing and related edge computing paradigms: A complete survey”. In: *Journal of Systems Architecture* (2019). DOI: 10.1016/j.sysarc.2019.02.009.

[43] PeiYun Zhang, MengChu Zhou, and Giancarlo Fortino. “Security and trust issues in Fog computing: A survey”. In: *Future Generation Computer Systems* 88 (2018), pp. 16–27. DOI: 10.1016/j.future.2018.05.008.

A Appendix

A.1 Literature Search Process

Scopus Search query:

```
TITLE-ABS-KEY ( fog AND computing AND privacy )
    AND ( LIMIT-TO ( PUBSTAGE , "final" ) )
    AND ( 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 )
    OR LIMIT-TO ( PUBYEAR , 2014 )
    OR LIMIT-TO ( PUBYEAR , 2013 )
    OR LIMIT-TO ( PUBYEAR , 2012 ) )
    AND ( LIMIT-TO ( SUBJAREA , "COMP" ) )
    AND ( LIMIT-TO ( LANGUAGE , "English" ) )
    AND ( LIMIT-TO ( SRCTYPE , "j" ) OR LIMIT-TO ( SRCTYPE , "p" ) )
```

A.2 Acronyms

BC Blockchain.

CC Cloud Computing.

EC Edge Computing.

FC Fog Computing.

FN Fog Node.

GDPR General Data Protection Regulation.

IDS Intrusion Detection Systems.

IoT Internet of Things.

MEC Mobile Edge Computing.

SDN Software Defined Networking.

TTP Trusted Third Party.

B Participation

Table 3: Participation Table.

Section	**Person(s)**
Introduction	Leonard, Lucien and Stijn
Methodology	Leonard and Stijn
Background	Leonard, Lucien and Stijn
Related work	Stijn
Privacy issues	Leonard, Lucien and Stijn
Privacy solutions	Leonard and Lucien
Synergies on Privacy	Leonard and Stijn
Discussion	Leonard and Stijn
Conclusion	Leonard and Stijn

Overall, with a few shifted priorities, work was evenly distributed.

Assignment 4: Literature Study

Web Services and Cloud Based System 2020
Lecturer: dr. Adam Belloum, Group 2

Hasine Efetürk
VU ID: 2527299
Vrije Universiteit
h.efeturk@student.vu.nl

Furong Guo
UvA ID: 12577790
Universiteit van Amsterdam
echoguo4826@gmail.com

Yaping Ren
UvA ID: 12985090
Universiteit van Amsterdam
yaping.ren@hotmail.com

June 1, 2020

Abstract

In this literature review, we explore the cycle of DevOps while focusing on the automation dimension of it. We investigate the impact of Infrastructure as Code on DevOps and the effective metrics for continuous delivery. Even so, we briefly discuss the impact of automation using IaC in relation to performance.

1 Introduction

For this review we will explore Infrastructure as Code, DevOps, metrics associated and the impact on business characteristics as latency and performance. This topic has evolved over the last decade and plays nowadays a crucial role in the business area of software delivery.

The outline of our literature review's structure is *thematic* with DevOps as a central theme identified where IaC and metrics are subsections that address different aspects of the central topic. The relevant literature which we collected, helped us to formalize our main- and sub research questions, as defined as follows

Main research question

What is the impact of Infrastructure as Code on DevOps and the defect prediction metrics required for a modern software delivery pipeline?

RQ1 What are the characteristics of DevOps?
RQ2 What are the characteristics of IaC?
RQ3 How can IaC contribute to the key features of Devops?
RQ4 How well can IaC perform using cloud service?

RQ5 How do we measure performance of DevOps and IaC?

Scope of literature stated coverage of our study and researched topics briefly, and give a summary on number and resource of the study. Then the next two chapters tried to answer sub-question 1 and 2 for features of DevOps and Infrastructure as code respectively. The following chapter gives a match-making on their features to show how is IaC important to DevOps. Chapter 6 discussed IaC deployment using cloud service to find it's advantage and limitation. Chapter 7 answers the last research question to give suggestions on how to measure and monitor DevOps and IaC in company. Finally in conclusion we gave a summary to research findings and presented possible topics of future work.

2 Scope of literature

This chapter describes the literature study to have a better illustration in coverage of our study. As explained in introduction, our research purpose will use DevOps as umbrella and find the benefit of application of infrastructure as code, and the metrics to measure the level of success.

Therefore we researched on the core feature of DevOps and its linkage to infrastructure as code, which is, how IaC could benefit implementation of DevOps and what is the expectation to have IaC. As IaC could be implemented using different services, we researched further on the application of cloud services to reveal its advantage and disadvantage to give more practical suggestions on implementation. As for metrics design, we explored the metrics to measure implementation quality and maturity level for DevOps. Certain factors which could be influenced by IaC are also studied to give a deeper evaluation on IaC.

The literature we studied includes core principle and pillars of each concept, together with real business cases. Because DevOps is not strictly framed, we would like to discover more in the best practices. As culture building is also of vital importance, the influence of landing of IaC to company culture is also discussed.

Our study coverage will be books, published papers, thesis, and web posts. And in total 20 references are used in this essay. Since the concept of DevOps is a set of practises and only raised 10 years ago, we will cover literature of all time but will focus more on the most up-to-date work. Also note that there are more practical findings on relatively smaller business on web posts, while the published paper might focus more on IT transition within a bigger organization. They might take different views in best practice of DevOps and IaC, therefore we adopted ideas from all resources as references.

In this literature study we will only cover materials written in English, study in other languages will not be in the research scope.

3 The Feature of DevOps

The general DevOps terminology encapsulates a combination of Development and Operations. The raise of DevOps is due to the growing need of software and IT industries. The growth of customers base and market outreach makes it essential that organizations internal delivery processes are optimized and in line

with business expectations. Time and resources are critical constraints in any business environment. Therefore, businesses are expected to acknowledge market needs in a short order whilst ensuring a higher level of quality. Consequently DevOps has emerged as a paradigm to bring innovative products and features faster to the market [1]. This is essential because, no organization can afford to live with manual, error prone and repeated activities in the software delivery lifecycle [2]. To emphasize, DevOps characterizes practises that streamline the software delivery process, learns by the feedback flow from production to development and improves the cycle time, which is the time from inception to delivery [3].

The movement for DevOps can be crystallized into five dimensions which all together characterize DevOps. The characteristics and dimensions of DevOps include collaboration, automation, culture, monitoring and measurement. In Lwakatare et al. [4] each dimension is explained in more detail. As stated in the introduction, our first research question is: **RQ1** *What are the characteristics of DevOps?* To answer this is question, a short description for each characteristic is covered in the following list.

1. *Collaboration*: rethinking and reorientation of roles and teams in development and operations activities
2. *Automation*: infrastructure and deployment process automation
3. *Culture*: empathy, support and good working environment for teams especially development and operations
4. *Monitoring*: Instrumenting application and aggregating monitored data into insights
5. *Measurement*: different metrics are used to monitor and assess the performance of processes in development and operations activities

The motivation for DevOps has its roots in the underlying conflict of development and operations teams which fulfill their core responsibilities in silos. This is typical for traditional IT where teams are divided by the type of work. Departments are dedicated only for writing code or solely for code testing. The incentives may depend on the creation of new features and server uptime / application response time for the development and operations department respectively. The conflict between the departments raise because the development department is in a need for change. They want to bring the changes and bug fixes to production fast. The second conflict is the fear of change of the operations team. They want to have a reliable solution for their end users with as little downtime as possible and where continuous feedback is provided and processed and continuous testing in production like environment is done.

To overcome those conflicts and break up the silos, the adoption of Agile methodologies become more important over time. For the Agile way of working and DevOps, there are certain stages in common. We identify five stages in figure. 1.

At the inception part, the system vision is developed, the project scope is defined and the business case is justified. The elaboration part is where the requirements are gathered and defined, risk factors are identified and system architecture is initialized. Subsequently, the software needs to be constructed,

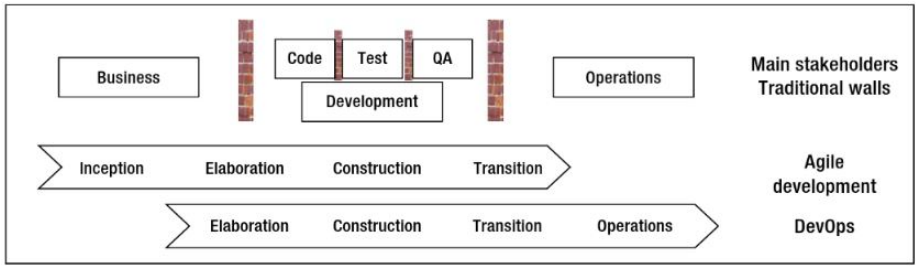


Figure 1: The differences and overlap for Traditional-, Agile- and DevOps methodologies

programmed and tested in the construction phase. When all the testing is done, the software is delivered to the user in the transition phase. After the software is live and available for the end user, the maintenance is done by the operations team in the operations phase [3].

The speed within a business environment is high and it is difficult for Dev/Test teams to adapt to the quick changes. DevOps allows you to do that by always having a prioritized product backlog, continuous channel of feedback with customers and ability to prioritize the product backlog all the time, directly taking business angle in consideration. There is a continuous process to plan small portion – execute - get feedback – react to feedback and adjust plan if needed and the cycle continues.

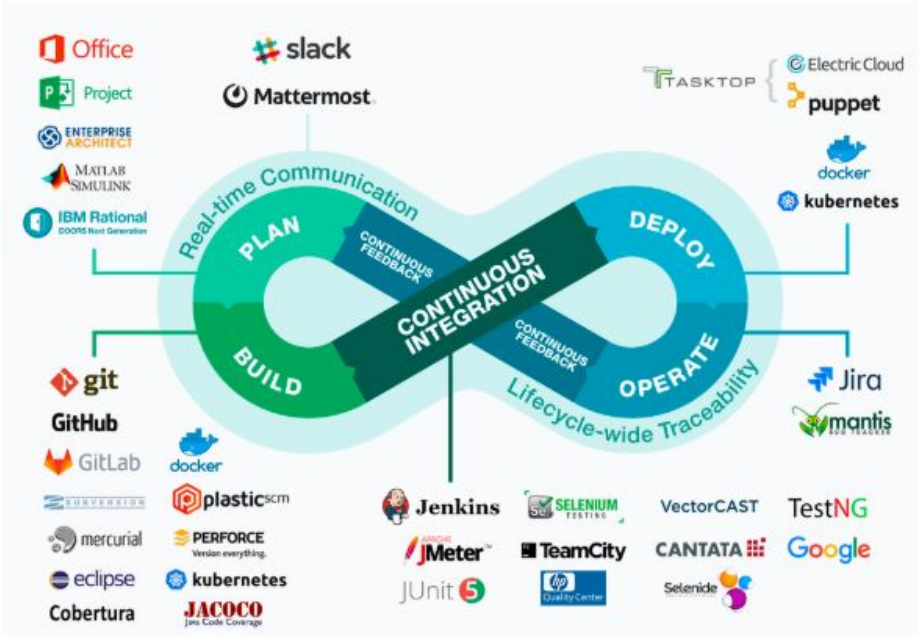


Figure 2: DevOps Cycle

In figure 2, different phases of the DevOps cycle are distinguished. *Continuous integration* refers to early integration where changes are not kept locally for a long period of time but instead are shared with team and validate how code behaves continuously. The process optimization stage is essential in achieving automation on a level, such that, as soon as the developer delivers changes, the build systems detects that. Then triggers a build carries out sanity tests and posts the build to a repository.

Proceeding, *continuous deployment* is the core part of DevOps and forms the critical piece of overall software delivery optimization. Surveys have shown that in majority of organizations the operations side of delivery is significant contributor to the delay in software delivery. Setting up the hardware to test the development build may take time varying from days to weeks. This manual deployment processes are inconsistent. DevOps principles recommend to automate the deployment and provisioning of hardware. Various cloud providers play a crucial role in this field. DevOps approach proposes that entire infrastructure provisioning should be maintained as code in source code repository where this concept is being called Infrastructure as a Code [2].

Prerequisite to *continuous testing* is to automate every test case. Any process that has to be repeated over time – should get automated, there are a lot of technologies available to meet that goal. Manual testing process must be evaluated for automation options and in majority of cases there will be ways to automate the same. This whole principle of continuous testing not only moves the testing process to early in cycle but also allows the tests to be carried out on production like system (complemented by continuous deployment) [2].

Continuous Monitoring is crucial for minimizing the downtime and optimizing the performance. As discussed in adoption approaches above, with the capability to test early on a production like system there is an opportunity to observe various quality parameters throughout and hence have the ability to react to any surprises in timely manner [2].

Lastly, as every good leader knows, you cannot improve what you do not measure, so measuring the software development process and DevOps transformations is more important than ever. As practitioners and professionals, it is known that software development and delivery is an increasingly difficult art and practice, and that managing and improving any process or system requires insights into that system. Therefore, measurement is paramount to creating an effective software value stream [5].

4 Infrastructure as Code

Infrastructure as code is a relatively new concept to automate management of IT infrastructure using modelled files to save human labor and improve deployment speed. In this section, we are going to discuss the advantages and limitations of IaC, approaches for implementation and brief introduction of available tools.

IaC provides possibility to automatically update system configuration on consistent and repeatable routines, which creates a faster and more reliable change management process [6]. Without the burden of monitoring system configuration changes, maintenance work is lighted for IT operation staff therefore providing several value adding points compared to traditional infrastructure solution. The reusable scripts of infrastructure code files would ease the knowl-

edge management process to further reduce the cost for new staff training. And the automated process enables a faster deployment cycle with much less human errors and security breaches [7].

There are two approaches for IaC programming, one is declarative and the other is imperative, where declarative approaches describes the logic of computations and are better at capture business processes while the imperative ones use statements more and could give better sequential information[8].

There are many tools which could support the implementation of IaC, here we only researched briefly on three of them for continuous configuration automation. We selected Puppet, Chef and CFEngine to make a comparison. These 3 tools are implemented in commonly used programming languages, but are using C++, Ruby and C respectively. Puppet and Chef are more operational friendly, while CFEngine is more develop friendly. Their learning curves are also different, the ope-friendly ones are more easy to learn while the other one cost more to train the staff although having faster speed and less dependency [9]. Although this paper did not cover much tools for comparison, here we only wanted to demonstrate the complexity and aspects to consider when choosing the IaC tool.

In the paper of Guerriero et al. [7] studying deployment and challenges of IaC from industry practical perspective, according to their conducted survey, there are concerns from both technical and management side. It would be very hard to test or guarantee system quality when there are a number of tools in use. More importantly, change of operational process and working culture could be a challenge worth consideration during IT transition.

There are more challenges on the technical side despite for the features of different tools during testing [10]. For example, the extra effort to implement an imperative script in an idempotent way, needs for script resources, execution issue for multiple instances together when using *Chef*, and dependency on external component.

While noting the benefits of adopting IaC, the features and challenges shall also be considered for real life application. Tool choosing could solve some of the issues, however IT should identify the root cause for a solution. For instance, if the business requirements are of high frequency and allow a slower but more stable deliver. Or system value stability and carefully code audit, the way to adopt IaC might need a second thought.

5 IaC within DevOps

This section aims to answer how value adding points and features of IaC matches with the principles of DevOps. The discussion is going to be from a technical aspect as well as a cultural aspect. After having a better understanding of why IaC is key to success of DevOps, we are also going to consider whether IaC is enough for infrastructure management within DevOps.

Technical link

The ability for companies to run micro services on agile infrastructure is made possible by IaC using configuration files based on text. This allows companies to better make use of containers and cloud based platforms, which much concern

eased for ability to scale in the future. Furthermore, the automated scripts also allows fast deployment with much administrative labor taken off. Perry [11] gives a nice summary on the ability of IaC, which are managed configuration via configuration file, instant deployment, automatic deployment, version control, best practice applicable directly to all systems and instant roll-back.

The core feature of IaC matches with DevOps in every way, therefore will not be further discussed here.

Cultural building

Function enabled by IaC also helps with culture building, which is of vital importance to the success of DevOps. To start with, the human efforts saved due to automation could release intelligent working staff from tedious routine work and devote more energy to more creative work. The degree of employee satisfaction and loyalty are measures which are very likely to increase accordingly.

Secondly, the fast and continuous delivery achieved by automated scripts supports smaller and more constant change. This increases the possibility of fail-fast [12]. Moreover, when the working process is integrated in time, communication within the team is also more smooth.

Thirdly, version control and ability to roll-back then encourages developers to go more creative and be more adventurous when the cost of fail is drastically lowered. What's more, this would also allow better project management when there is a development conflict. The more prioritized business needs could get pass first easily with fast roll-back.

Is IaC enough for DevOps

Mark Robinson marked on IaC that only having the automation could not solve everything, operators are still required for better understanding of architecture design and technical process. A technology without human supervision still might go wrong. Another point that needs attention is that even though we are trying to combine development and operation together, they should not be considered as the same. IaC gives the chance to a fully scalable dynamic infrastructure completely configured by code [13].

Comparison made in this chapter is more based on researches and summaries from previous two chapters. An automatic deployment system needs to be established for DevOps to work [14], and IaC is inseparable with this automation. Technically IaC provides possibility to achieve agile goal of DevOps, at the same time, company culture can also be changed with this strong instrument. However researches on whether DevOps must go with IaC is not available, therefore not included in this essay.

6 IaC on Cloud and Performance

Deployment pain is a measure of fear and anxiety that engineers and technical staff feel when they push code into production [15]. It also measures the extent to which deployments are disruptive rather than easy and pain-free. Where deployments are most painful, you'll find the poorest software delivery performance, organizational performance, and organizational culture.

Teams can reduce deployment pain by implementing key technical capabilities. Those may vary from implementing comprehensive test and deployment automation, use of continuous integration, including trunk-based development, shift left on security, effectively manage test data, use loosely coupled architectures, ability to work independently, and use version control of everything required to reproduce production environments decrease their deployment pain [15]. As a result, the technical practices that improve our ability to deliver software with both speed and stability also reduce the stress and anxiety associated with pushing code into production.

The speed within a business environment is high and it is difficult for Dev/Test teams to adapt to the quick changes. DevOps allows you to do that by always having a prioritized product backlog, continuous channel of feedback with customers and ability to prioritize the product backlog all the time, directly taking business angle in consideration. There is a continuous process to plan small portion – execute - get feedback – react to feedback and adjust plan if needed and the cycle continues.

This chapter is to form an answer to the fifth research question: **RQ5** How do we measure performance of DevOps and IaC? In the next section, we will dive further into the technical capability of implementing test and deployment automation with the usage of state-of-the-art tools to achieve a better performance.

Performance Optimization with Tools

Continuous delivery is on a large scale enabled by the usage of tools. Tools are mandatory in automating DevOps. Quality deliveries with short cycle time need a high degree of automation [1]. There are a lot of tools to test and deploy code based on your infrastructure. To enhance the business performance, the right choice of deployment automation tool is crucial to enable a reliable IaC. The various topologies needed to be tested and create requiring deployment patterns may be identified by developers. As well as to automate the consecutive steps related to installation of application stack on the cloud provisioned image, populating it with test data, triggering the automated test suite and pushing the results to a central repository.

The choice of the right tool is based on certain criteria set for each DevOps phase. The build and deployment phases are crucial for a right IaC script. In the build DevOps phase, the tools must support fast workflows. Build tools help achieve fast iteration, reducing manual time-consuming tasks. DevOps pushes automation from the application to the infrastructure. Compared with manual infrastructure provisioning, configuration management tools can reduce production provisioning and configuration maintenance complexity while enabling recreation of the production system on the development machines. Such tools are a major DevOps enabler, especially as architecture moves from a monolithic block of software to a microservices approach[1].

Although researchers have focused considerably on build and deployment to achieve a better performance, DevOps also impacts infrastructure operations. So, you need tools to maintain your infrastructure's stability while ensuring high performance [1]. For the operational phase of DevOps, the tools could be divided into logging and monitoring types.

Cloud Benchmarking

Another key method to optimize the business performance while diving into IaC is to have a proper cloud based resource provider. Cloud WorkBench was designed and implemented to leverage the notion of IaC for cloud benchmarking, and is used to automate the benchmarking lifecycle from the definition to the execution of benchmarks. Its extensibility allows to add additional benchmarks at runtime and support new cloud providers with minimal effort. CWB is used to execute extensive benchmarks over different cloud providers [16].

Cloud WorkBench differentiates from these other approaches via its strong IaC core, which makes it easy to define benchmarks based on standard tooling and concepts, as well as share benchmark definitions. Further, CWB is one of the few approaches which is known to offer provisioning capabilities. There is also no solution known that integrates periodic scheduling functionality into a web-based framework. Furthermore, CWB together with CloudBench are the only frameworks designed for benchmark extensibility at runtime [16].

7 Metrics design

As we described in previous chapters, the main features of DevOps including automation and continuous delivery. Like agile methodology, it encourages changes incremental and faster. And most importantly, it should serve for a core purpose - higher quality and business satisfaction. Together with the help of IaC, how do we know the actual impact of having DevOps and IaC running in business? How do we measure the mature level of each aspect and how to measure them? As DeMarco supported, we can not control what we can not measure [17]. In this chapter we are going to firstly discuss good metrics for DevOps. Because there are not much direct study on IaC quality metrics, and knowing the tight relationship between IaC and DevOps, we are going to discuss on how to put the vital DevOps metrics to use for measuring IaC.

In this book [18] introducing continuously delivery and DevOps, there are some simple but effective metrics for CD.Mean time between failure (MTBF), for the frequency of issued in system. This could help to measure the stability and overall quality of system. Mean time to resolution (MTTR) measures the speed of bug fixing. Bug escape distance measures the time to locate the issue. Tother with other metrics such as code complexity, unit test coverage and commit rates, they construct the core quality metrics of DevOps. The key factors identified here are effective but might be too simple and general for practice use. As there are different phases in DevOps pipeline, there should be more detailed or targeted metrics raised for specific SLA monitoring.

While Cusick [19] also identified MTTR and MTBF plus Mean Time to Failure(MTTF) as classic measurements to system stability in change management. This paper focus more on IT transition from traditional ITIL driven process to DevOps, but also needs measurements to keep track of SLAs and uses CT cooperation as an example. Since IaC goes tightly with change management in DevOps, the metrics identified here could be a stronger reference to build metrics towards IaC specifically.

There are a lot more metrics which could be categorized to four major aspects, which are velocity, quality, performance and satisfaction [20]. It noted

that also the traditional metrics are also applicable, it's worth notice that due to the quick and incremental feature of DevOps, time to detect and fix the failures could be longer when there are multiple changed at the same time. This would make MTBF a useless metrics.

Another set of DevOps metrics are introduced by Forsgren and Kersten[5] is having system-based metrics and survey-based metrics as complementary as each other. Using survey measures could give an overview for the value stream to avoid great gap between IT and business value, while system measures could make full use of automatically collected tool-based data to make a continuous view. This set of metrics, like Cusick, also pay much attention to IT transformation from traditional one to DevOps, therefore their metrics choice would either focus on keep previous SLAs under monitoring for less shock to working staff for a more smooth transition, or on carefully and continuous system status monitoring to avoid major deviation. At the same time, online forums and blogs like Devopedia and Thoughtworks could be giving suggestions to teams starting under DevOps with relatively simple business cases therefore does not require transition phase.

IaC as part of DevOps it can naturally inherit the metrics for certain monitoring purpose, the quality metrics could be monitored using system-based metrics [5] where data could be generated and correlated for further analysis to monitor their performance under SLA. However, the choice of metrics should also be very much rely on the actually need of business and mature level of IT management. Moreover, combining with the challenges discussed in chapter 4, there shall be metrics targeting weak points identified with consideration of tools under use.

8 Discussion and Conclusion

The review is addressing a *methodology* combined with *quantitative research* because the implementation of IaC is dependent on certain business requirements where each business has a roadmap to enable their vision statement. Combining this from several business areas qualifies for quantitative research.

We used five sub-research questions to support answering the main research question: What is the impact of Infrastructure as Code on DevOps and the defect prediction metrics required for a modern software delivery pipeline?

To sum up, Infrastructure as Code is the automation characteristic which enables a faster time to market, enhances the working culture, enables performance optimization and adapt to continuous feedback to improve and streamline the software delivery process. Which matches with the mindset of DevOps in all sorts of ways not only technically but also culturally.

IaC metrics could take DevOps metrics as references, there are several set of metrics we can use, but remember the choice of metrics should also rely on the real need of business and the maturity level of IT management

Further research could focus on the analysis of DevOps tools and how they could interoperate to enable an optimal performance.

Section	Author
Introduction	All
Scope of Literature	Furong
The Feature of DevOps	Hasine
Infrastructure as a Code	Yaping
IaC within DevOps	Yaping
IaC on Cloud and Performance	Hasine
Metrics Design	Furong
Discussion & Conclusion	All

9 Work distribution

References

[1] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3):94–100, 2016.

[2] Manish Virmani. Understanding devops & bridging the gap from continuous integration to continuous delivery. In *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, pages 78–82. IEEE, 2015.

[3] Michael Hüttermann. *DevOps for developers*. Apress, 2012.

[4] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. An exploratory study of devops extending the dimensions of devops with practices. *ICSEA 2016*, 104, 2016.

[5] Nicole Forsgren and Mik Kersten. Devops metrics. *Communications of the ACM*, 61(4):44–48, 2018.

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

[7] 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. IEEE.

[8] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In *Enterprise, Business-Process and Information Systems Modeling*, pages 353–366. Springer, 2009.

[9] Clauirton Siebra, Rosberg Lacerda, Italo Cerqueira, Jonysberg P Quintino, Fabiana Florentin, Fabio QB da Silva, and Andre LM Santos. From theory to practice: the challenges of a devops infrastructure as code implementation. In *Proceedings of the 13 th International Conference on Software Technologies (ICSOFT)*, 2018.

[10] Waldemar Hummer, Florian Rosenberg, Fábio Oliveira, and Tamar Eilam. Testing idempotence for infrastructure as code. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 368–388. Springer, 2013.

[11] Yifat Perry. Why you need infrastructure as code to do devops properly, Jan 2020.

[12] Matt Callanan and Alexandra Spillane. Devops: making it easy to do the right thing. *Ieee Software*, 33(3):53–59, 2016.

[13] Team Cloudify. Infrastructure as code - is it really enough for devops?, Jan 2020.

[14] Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[15] Jez Humble and Gene Kim. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution, 2018.

[16] Joel Scheuner, Philipp Leitner, Jürgen Cito, and Harald Gall. Cloud work bench–infrastructure-as-code based cloud benchmarking. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 246–253. IEEE, 2014.

[17] Tom DeMarco. *Controlling software projects: management, measurement & estimation*, volume 1133. Yourdon Press New York, 1982.

[18] Paul Swartout. *Continuous delivery and DevOps: A quickstart guide*. Packt Publishing Ltd, 2012.

[19] James J Cusick. Achieving and managing availability slas with itil driven processes, devops, and workflow tools. *arXiv preprint arXiv:1705.04906*, 2017.

[20] lokesh.rawat arvindpdmn. Devops metrics, sep 2019.

A review on parallel implementations of DNNs based on distributed algorithms

Dongqi PU
Vrije Universiteit Amsterdam
Universiteit van Amsterdam
dongqi.pu@gmail.com

Carlijn Nijhuis
Vrije Universiteit Amsterdam
Universiteit van Amsterdam
c.e.nijhuis@student.vu.nl

Juan Agustin Tibaldo
Vrije Universiteit Amsterdam
Universiteit van Amsterdam
tino@ohzi.io

Abstract

In recent years, the research of deep neural networks has been accompanied by its successful applications in speech recognition, image recognition, and natural language processing, and has become one of the most hot research topics in the field of artificial intelligence and machine learning. The rapid development of neural networks in big data applications has also promoted the development of parallel processing devices and platforms, including the currently widely used parallel processing platforms such as Hadoop and Spark, and GPU (Graphics Processing Unit) hardware devices with general computing capabilities. This article first introduces the background of deep neural networks and commonly used calculation models, and then compares and analyzes the commonly used deep neural network open source software from the perspective of supporting hardware, parallel interface, and parallel mode. Finally, the future development trends and challenges of parallel deep neural networks are prospected.

1 Introduction

Today, deep learning has made breakthroughs in many fields[1] and has been widely used, in areas such as speech processing, computer vision, natural language processing, man-machine games, and autonomous driving. Here they have achieved unprecedented results. Because deep learning needs to constantly derive and iteratively update the model to improve its performance, it requires a lot of calculations, and is a typical calculation-intensive task. Therefore, the training process of these neural networks is very time-consuming. As the number of data and model parameters increases, the growth rate of the stand-alone memory (or video memory) will not match it. Therefore, deep learning training by a single node can no longer meet the requirements. Distributed deep learning has become an effective method to solve this problem. Due to its good flexibility and scalability, distributed networks effectively combine stand-alone resources. In recent years, many researchers and companies have realized the importance of distributed deep learning and have begun to perform research in this area, aiming to make full use of distributed clusters and clouds to efficiently train neural network models.

The aim of this paper is to give an overview of the current state of the research on distributing deep neural networks on the cloud and to identify the problems that still need to be solved, specifying the road that future research should take.

This article first briefly explains the theory behind deep neural networks (DNN). Then it gives an overview of the different types of frameworks and open source software for creating and training DNNs. Before specifying the current techniques to parallelize and distribute DNNs on the cloud through data and model parallelism and the performance gained through these methods. Finally, the current challenges and prospects of the research area are discussed.

2 Deep Learning

Deep learning[2] originated from the simulation of the human brain, using logical nodes to represent neurons, and simulating the input and output of neurons through weights. People first created a multilayer perceptron (MLP) based on the basic perceptron and combined artificial neurons to fit complex functions through simple calculations. The hope was that the neural network model could be used to extract information at a more abstract level. A model contains of an input layer, an output layer, and multiple hidden layers. As shown in Figure 1, data is passed to the input layer in the form of vectors, and some operations such as vector matrix calculation and activation function are passed down layer by layer, and finally the output result is obtained at the output layer. The deep neural network model can theoretically be fitted to replace any complex function, and has a strong representation ability.

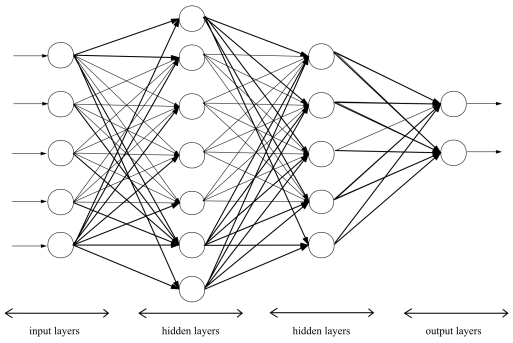


Figure 1: Fully Connected Neural Network

Compared with traditional machine learning algorithms (LR, SVM, Decision Tree, Random Forest), the deep learning method can automatically extract high-level features from the input data, such as the contour and shape of the image, through this type of network model. There is no need for data scientists to do feature engineering. On this basis, the introduction of complex network models makes deep learning have the ability to deal with spatial dependence (overall and local) and time dependence (causal relationship) problems.

3 Parallel programming and neural network framework

This section will introduce the most common parallel programming frameworks and open source systems for distributed deep neural networks.

3.1 Parallel programming framework

1) CUDA[3] is a parallel programming language, which was launched by NVIDIA in 2007 and can only run on various types of GPUs of the company. It uses extended C language for GPU programming. Since the birth of CUDA version 1.0 in 2007, due to greatly reduced difficulty of GPU general programming, a large number of researchers have tried to use GPU to accelerate algorithms in various fields. Since then, the CUDA version has iterated rapidly, and the general computing power has become stronger. The CUDA parallel programming model uses a two-level parallel mechanism, that is, Block is mapped to stream multiprocessor parallel execution and Thread in the same Block is mapped to stream multiprocessor CUDA core for concurrent execution.

2) OpenCL[4] is a unified programming standard for heterogeneous computing developed by the Khronos organization, which is supported by companies such as AMD, Apple, Intel, NVIDIA, TI and so on. It can run on multi-core CPUs, GPUs, DSPs, FPGAs and heterogeneous acceleration on the processing unit. When the OpenCL calculation model is executed on specific hardware, the operating environment of each manufacturer has responsible for compiling the code into machine code and establishing a hardware and software mapping mechanism. When the core kernel executed by OpenCL is started, a large number of threads will be created to execute at the same time, and each thread is a work-item to complete the operation defined by the kernel function. When mapped to OpenCL hardware for execution, a two-level parallel mechanism is used. Work-groups run concurrently on computing units of heterogeneous computing devices. Multiple work-items in the same work-group are independent of each other.

3) OpenMP[5] is a parallel programming model based on shared memory and multi-threading. When using it, programmers need to add keywords for parallel compilation in the part of the program that can be used for parallelism. The operating environment maps the computing tasks to multi-threaded parallel execution based on the keywords. The OpenMP parallel technology has good portability, does not require a large number of modifications to the serial code, has a strong flexibility, and can be easily adapted to different parallel system configurations. The thread scheduling for executing parallel tasks is controlled by the OpenMP operating environment, so when the number of threads is large, the scalability of parallel programs [6] often behaves poor.

4) MPI is a parallel programming based on message passing framework [7]. MPI is mainly used in cluster computing environments. As a cross-language communication protocol, MPI supports both point-to-point and broadcast communication methods. When the MPI program is executed, multiple processes are started on each node of the cluster. The processes between the nodes explicitly exchange messages through high-speed communication links (such as Ethernet or InfiniBand), and work together in parallel to complete computing tasks. MPI is widely used in the high-performance computing industry, but parallel programs based on MPI usually have large changes in algorithms, difficult programming, and insufficient fault tolerance. If a process fails, the entire application needs to be recalculated.

5) Spark[8] is a universal parallel computing framework. It extends the widely used MapReduce computing model, and adds interactive query and stream processing functions. It also supports Scala, Java, and Python languages. It is easy to use and can work with multiple frameworks such as Yarn, Mesos, Hive, HBase, HDFS, etc. Spark abstracts all types of data structures into RDD structures. Based on the characteristics of in-memory computing, it has obvious advantages in iterative algorithms compared to Hadoop, and the efficient fault tolerance mechanism enables it to return to normal in the face of failure problems. At present, Spark has been widely used in the field of big data processing.

Table 1: Comparison of parallel programming frameworks

Framework	Announcement date	Openness	Main supported languages	Supported hardware	Programming difficulty	Supported Deep learning library
CUDA	2007	Private	C, C++	NVIDIA GPU	Easy	Caffe, TensorFlow, MXNet, CNTK, (Py)Torch, Theano
OpenCL	2008	API	C, C++	GPU, CPU FPGA, DSP MIC	Difficult	Caffe, Theano
OpenMP	1997	API	C, C++, Fortran	CPU, MIC	Easy	Theano
MPI	1992	API	C, C++, Fortran	CPU, MIC	Difficult	CNTK, S-Caffe
Spark	2010	Open source	Java, Scala, Python	CPU	Easy	Intel BigDL, SparkNet

Table 1 analyzes and compares the parallel programming frameworks in terms of openness, development difficulty, and support for deep learning software.

3.2 Neural network open source software system

Due to the wide application of deep neural networks in various fields, industry and academia have launched relevant open source software systems, so that researchers can quickly apply deep neural networks to their own research fields. At present, the open source software systems of deep neural networks mainly include: Caffe, TensorFlow, MXNet, CNTK, PyTorch, Theano, etc. as shown in Table 2 These use some similar methods for parallel deep neural networks[9], for instance, use high-performance multi-threaded libraries on the CPU to train the network, and support cuDNN libraries on the GPU to accelerate the network. In the implementation of distributed parallel deep neural networks, data parallelism or model parallelism is used.

Table 2: Comparison of open source libraries for neural networks

Framework	Announcement date	Supported hardware	Parallel interface	Parallel mechanism	Multi node
Caffe	2013	CPU, GPU	CUDA, OpenCL	Data parallelism	Support
Caffe2	2017	CPU, GPU	CUDA, OpenCL	Data parallelism	Support
TensorFlow	2015	CPU, GPU, MIC	CUDA	Data parallelism, model parallelism	Support
TensorFlow2	2019	CPU, GPU, MIC	CUDA	Data parallelism, model parallelism	Support
(Py)Torch	2017	CPU, GPU	CUDA	Data parallelism, model parallelism	No support
Theano	2007	CPU, GPU	CUDA, OpenCL	Data parallelism	No support
MXNet	2015	CPU, GPU	CUDA	Data parallelism, model parallelism	No support
CNTK	2016	CPU, GPU	CUDA	Data parallelism	Support

(1) Caffe[10] supports hardware platforms as CPU and GPU. It uses CPU's high-performance multithreading libraries (Altas, OpenBLAS, Intel MKL) to train the network; when using GPU, Caffe uses cuDNN for training. Data parallelism is used to accelerate the deep neural network, achieving single-node multi-device data parallelism. The tree topology connection strategy is used to exchange parameters between multiple GPUs, which can effectively reduce communication overhead.

(2) TensorFlow[11] uses computation graphs to represent machine learning algorithms. To parallelize, the user first needs to create an abstract TensorFlow cluster object, which is used to execute the computation graph in a distributed manner. TensorFlow supports two parallel modes for model and data. For model parallelism, one must define the structure of the computation graph, the configuration of the cluster, and map the different parts of the model on different computing devices. Each abstract cluster includes multiple tasks, and each task represents a computing task of the deep neural network. The user maps the task on different devices. For data parallelism, an abstract cluster can be divided into multiple jobs. A job is generally divided into a worker job and a parameter job, and a job contains multiple tasks. In TensorFlow, the worker job is responsible for independent training of a deep neural network on each node, and the parameter job is responsible for the function of parameter exchange and gradient update.

(3) MXNet[12] realizes parameter update and data exchange through KVStore, and implements resource scheduling and task management through the engine. KVStore is a distributed key-value store based on the parameter server, used for data exchange between multiple devices. The engine is a resource management and task scheduler, which is used to schedule the execution of KVStore's operation and management parameter update mechanism. MXNet adopts a data parallel strategy with a two-layer architecture. The first layer is to use a parameter server for data parallelism among multiple devices in a working node, and the second layer is data parallelism between different working nodes. When training deep neural networks, the parameters on multiple devices of a single node are first summarized, and then the single node sends the data to the parameter server for parameter update. Since the communication bandwidth within a single node is much larger than the communication bandwidth between nodes, the two-level parallel strategy can effectively reduce the bandwidth requirements.

(4) CNTK[13] represents a neural network as a series of calculation steps through a directed graph. The leaf nodes of the directed graph represent input values or network parameters, and the other nodes represent matrix operations on their inputs. CNTK uses a data parallel method to implement a parameter server that supports synchronous and asynchronous parameter update mechanisms. At the same time, CNTK can be deployed on heterogeneous clusters on multiple GPUs or CPUs for distributed training of deep neural networks.

(5) Torch[14] and Torch-based PyTorch support multi-GPU and CPU training. Use OpenMP, Intel MKL and Pthread on the CPU for multi-thread training, and support cuDNN and OpenCL on the GPU. Torch supports two strategies of data parallelism and model parallelism. For data parallelism, a model copy is independently trained on each device. When the parameter is updated, the gradient on each device is transferred to the parameter device for parameter update. Model parallelism is to copy the training data to all submodules and run the disjoint parts of the model on different devices.

(6) Theano[15] uses symbolic graphs to describe machine learning algorithms and supports CPU and GPU for network training. Theano uses OpenMP on the CPU for multi-thread training and cuDNN on GPU. It adopts a data parallel method and supports a multi-GPU synchronization parameter update mechanism.

The open source software for deep learning is updated quickly, and the supported hardware architecture and software platforms are also increasing. In particular, support for embedded platforms[16]

and Python has become a key development direction. Table 2 compares these open source deep learning frameworks.

4 DNN on the cloud and edge

4.1 Introduction

There is enough motivation to promote the development of distributed deep neural networks (DDNNs) in the cloud and the edge. Recently, there has been an exponential increase in the needs of image recognition due to the universal use of small devices, namely, mobile phones and sensors. Another drive for this increase in the research in these topics is the need for an increase in speed-up typical of HPC, which has motivated research on this topic. Although HCP has always had a bottleneck in memory allocation and inter-computing unit communication, the former requires a smart way to split the DNN algorithm between the devices, the edge and the cloud and, the latter requires high bandwidth and good memory allocation and computer power allocation. In the following sections, we will focus on DDNNs in the cloud and the edge.

4.2 Motivation

There are four major points that motivate the usage of DNN on the cloud:

IoT, increase of techniques using Neural Networks, and high connectivity
There is an evenly increasing number of devices connecting to the Internet every day. Mobile phones have been mainstream for years, but recently new devices are getting attached to the internet. These are new types of sensors, appliances such as fridges and washing machines.[17] These devices generate all kinds of information, and recently they have been requiring the use of Neural Networks to provide features that were not there before, such as automatic detection of fridge content and so. Nevertheless, the most common data to be shared on the internet is visual data, namely, video. Video has been an 80% of a staggering 250 exabytes of data that has been generated in 2018. This is due to several factors: mobile phone supporting large 4k video, video streaming platform adoption, IP cameras, and the 24/7 use of mobile phones that are no longer bounded to stay at home to be connected.[18, 19, 20, 21, 22] This increase of visual data is demanding an evenly increase in the use of visual recognition. This image analysis requires almost real-time latency, as there is either a stream of information (IP cameras for example), or there is an end-user waiting in the prediction pipeline, suffering from communication latency and possible quick drainage of battery power due to such an intense communication. [19]

DNN usage for Computer Vision
DNN algorithms have become the de facto method of doing computer vision[18]. The current state of the art for visual recognition included DNN and convolutional NN algorithms, that are designed with more and more layers and filters as the time pass. They have become more powerful and accurate, with the cost of more computing power. This means a device, or the edge, might not be able to fully run the complete DNN algorithm [21].

Advantages of NN processing in the device and edge
Devices have a limited amount of computing power and memory available. This might not allow them to run a whole DNN model, nevertheless, they can, in some cases, execute some layers of the pipeline. The same can be said about the Edge. Having layers being executed before reaching the cloud have clear benefits: reduces the memory bandwidth usage (as there might not be a need for sharing the whole dataset but only a summary of it), reduces the computing power required on the cloud, and thus reducing its saturation, reduces latency as all the NN might be run in the device/edge and, privacy, as it is not required to send raw data that could eventually include sensitive information[21]

Budget
Another thing to note is the fact that a neural network that is fully run on the cloud will require a larger budget when the service is fully running and providing to a large audience. This is related to fees cloud providers charge, as most of their services are charged on demand, then the more computation done on the cloud the more expensive the service is going to be. In some cases, this might be prohibitive or just high enough to think about a different approach. In this context budget learning is a concept where the learner tries to use the cloud services as less as possible, based on the economic constraints.[19]

4.3 Infrastructure Stack

Typical HPC vs Cloud/Edge/Device architecture

A Typical DNN architecture is GPU based, has a predefined network interconnection such as high-speed InfiniBand connection, estimated power consumption, and refrigeration needs, thus creating a fixed set of capabilities.[23, 24] In contrast, Cloud/edge based Neural Networks have a heterogeneous architecture composed of devices of various brands, power consumption, power autonomy, protocols, and computing power. Most of the time we see the same type of device used on the edge, and device layer, but this is not always the case (IP camera, remote sensors)[20]. This means the code needs to adapt to the underlying architecture. Another important factor is that this architecture uses the internet for communication layer, being the internet prone to speed and availability fluctuation and even eventual disconnection, so the architecture needs to be redundant and tolerant to these events. [20]

Describing the architecture

The infrastructure of the Cloud and Edge Neural Network is composed of three players: the devices, the edge/fog, and the cloud. The cloud has been described many times as an on-demand, globally distributed, with rather uniform latency, convenient, limitless having a pay-as-you-go profitable structure. They are usually in the middle of big cities and are distributed through the internet by routers and elements on the edge of the internet. Most of the time it is a cost-efficient alternative to run a local HPC system.[25] Devices are a heterogeneous group of computers, phones, appliances, sensors, and so, that are connected to a router and sits on the very end of the network structure, where the end-user also is.[25] Finally, the edge is what sits at the end of the network before the end device. The fog is characterized for having low latency and location awareness, for being wide-spread geographically located, for allowing mobility, for having a very large number of nodes, for having a predominant role in wireless access, having a strong presence of streaming and real-time applications and also has wide heterogeneity.[25]

4.4 Cloud algorithm techniques

A GPU is great for DNN, but has clear limitations when the whole dataset does not fit into memory. To scale these techniques, distributing algorithms can be used. However, creating a distributed algorithm for NN using regular techniques also presents a challenge, as the more nodes there are, the bigger the communication cost is.

DistBelief [27] was proposed as a software architecture for handling big datasets on DNN on distributed clusters. It takes care of communication and synchronization. DistBelief is able to run both on a single device as a multi-threaded application or on a cluttered distributed architecture. It uses message passing for communication. It also implements two techniques for running this type of NN: **Downpour SGD** implements a distributed version of stochastic gradient descent, a technique that is mostly used on the contract of DNN. This tool implements a centralized server that centralized the management of parameters, updating their gradients depending on the gradients generated by the clusters i.e. the model replicas. These techniques also exploit the Adagrad technique, which allows adaptive learning rates for the parameter, increasing the overall speedup. **Sandblaster L-BFGS** is another technique that allows NN implementation using L-BFGS on very large models. It does not use a centralized parameter pool, but a coordinator, and it is effective on a low bandwidth system. [27].

The DistBelief software was proposed for Google Brain and eventually became TensorFlow. [27]. TensorFlow was designed as 'a flexible data flow-based programming model'[27] capable of handling the environment where Cloud DNN lives.

4.5 Tools for distributing algorithms on the cloud and the edge

Couper
A tool that has been proposed by one of the authors is Couper [18], this tool allows the model to be

sliced and to be run efficiently at the edge infrastructure in a unit called a 'slice'. This process can be done automatically, best fitting the underlying architecture of the application. Couper can be applied with arbitrary production models and for different infrastructure configurations. It also provides tools for adapting to different engines and let the developer tune different parameters in order to create the ideal slicing for the current application. The end result is a deployment-ready DNN pipeline that has demonstrated a notable speedup when compared with only cloud or only edge models. This tool finds breaking points where the NN can be split. These slices can be run on parallel in the case of filters that can be run in parallel. The tool is able to analyze the network, the memory usage and, possible drops in latency and modify the breaking points by improving itself. The slices are run using Kubernetes, and a small overhead for serialization [18].

Containerized DNN
A containerized partition-based runtime adaptive convolutional neural network (CNN) acceleration framework for the Internet of Things (IoT) has also been proposed [20]. This framework uses spatial partitioning techniques and layer fusion, creating optimal partitions in a dynamic fashion. These partitions are created depending on the computing power available for the application and also depending on the network conditions. Partitions on this application are then containerized, creating an abstraction that is easy to run no matter what the underlying platform is. This technique uses Docker and Kubernetes to handle the resources and scheduling of containers these containers. This Kubernetes wraps one or many layers, and can eventually be run on parallel. [20]

Bracketing
Bracketing is also a proposed technique for dealing with cloud DNN applications [19]. It allows the architecture to handle the entirety of the machine learning algorithm up to a point where the accuracy is lower than a point. If the accuracy drops too much, the NN execution is done by a bigger machine located on the next layer.[19]

5 Performance Analysis

Now that the most common different approaches and frameworks to distribute deep neural networks on the cloud have been discussed, this section will provide an overview of the different performance analyses that have been performed on distributed deep neural networks, mentioning the bottlenecks that were discovered and proposed ways to overcome them.

5.1 Analysis

The paper by Dean et al [26] set the precedent for further investigation on how to distribute deep neural networks. In their paper, they mention that larger models can dramatically improve performance and hence it is beneficial to look for ways to efficiently train models with billions of parameters. Therefore, they propose a software framework called 'DistBelief' that can be used on a cluster with thousands of machines to train large deep neural networks. As already mentioned in section 4.4, the framework consists of two algorithms: 'Downpour SGD' and 'Sandblaster'. Downpour SGD is an asynchronous data parallel procedure for stochastic gradient descent (SGD), which is the prominent method used to train deep neural networks. Sandblaster is a framework for batch optimizations, like the distributed implementation L-BFGS. Batch is a term used for stochastic gradient descent, where batches are used to make the algorithm data parallel, where each node gets a part of the training sample called a minibatch to train upon. They claim that their system is capable of training a deep neural network that is 30 times larger than tried in previous literature and therefore also achieves state-of-the art performance on ImageNet, a common metric benchmark for visual recognition tasks. Furthermore, it also accelerates the training time of speech recognition tasks.

Although the results of Dean et al [26] were very promising, especially their Downpour SGD in combination with AdaGrad, Seide et al [28] analyze the theoretical efficiency of model and data parallel SGD. Their conclusion is that the methods do not scale well to more nodes, running into a bandwidth bottleneck. Important to mention is that Seide et al [28] focus on parallelizing the plain SGD method through model and data parallelism, as opposed to the asynchronous variant proposed by Dean et al [26]. They made their conclusion by analyzing the theoretical upper bound of the parallelizability of SGD, estimating the optimal number of compute nodes to maximize the efficiency. Seide et al [28] claim that any optimal version of data parallelism entails some form

of delayed update as is the case with the Downpour SGD method from Dean et al [26], where the nodes continue training their version of the model, while the results from previous iterations are still being communicated across all the nodes. According to them, the optimal training is where the computation and the communication happens concurrently at the exact same time. A version of this is Asynchronous SGD (ASGD), but they say that ASGD does not improve the parallelizability much, since, they say, if deterministic data-buffered data parallelism does not scale well, neither would ASGD. Their explanation and reasoning for this statement is lacking, not explaining why they think that ASGD is also linked to the theoretical bound they put on plain SGD, effectively eliminating any further algorithmic optimization for SGD, saying that in essence SGD is not very good to parallelize. Furthermore, in the experimental phase, where Seide et al [28] try to prove their theories through experimentation, they mention that they cannot provide time measurements for data parallelism, because of lacking an optimized implementation. Therefore, it is only through theoretical statements that they make that claim, claiming that the best way to speed-up the training process is to inherently change the training algorithm to allow for greater parallelizability. There is truth to this however, due to the inherent sequential nature of SGD. Seide et al [28] concur that the efficiency of SGD can be improved by increasing the minibatch size and compressing the data. Compressing the data is a method that is analyzed in other papers that will be mentioned later in this section.

Increasing the minibatch size is contradicted by the paper by Keskar et al [29]. Keskar et al [29] say that increasing the minibatch size leads to a poorer generalization of the model. This is already touched upon by Seide et al [28] by them mentioning that the increase of the minibatch size is limited by the training stability. Keskar et al [29] investigate the cause of the poorer generalization and they come to conclusion that it is due to the fact that large-batch methods tend to converge to a sharp minimizer, while small-batch methods tend to converge to flat minimizers, meaning that they generalize better due to the fact that they can be specified with lower precision. The reason that small-batch methods converge to flat minimizers is explained as being due to the noise in the gradient estimation of small-batch methods, which is greater than for large-batch methods. This noise allows small-batch methods to escape the sharp minimizers, as it pushes the iterates out of the minimizers, encouraging movement towards flat minimizers. The noise has less effect with flatter minimizers. Keskar et al [29] agree with Seide et al [28] that the speed-up and scalability is limited by the minibatch sizes. However, increasing the batch size is, according to them, not suitable. They propose some possible solutions, like data augmentation, conservative training and robust optimization, but these do not really solve the problem. As an alternative they mention the option of dynamic sampling, where the batch size is increased gradually after each iteration.

Keuper et al [30] also proceed to analyze the performance from a theoretical point of view. They begin by specifying the main bottlenecks, according to them, for scaling distributed deep neural networks. Keuper et al [30] present fixed theoretic constraints that explain the lack of proper scaling beyond a couple of nodes. The bottlenecks being model distribution overhead (communication), data parallelized matrix multiplication and training data distribution. The mentioned perfect overlap of communication and computation by Seide et al [28] is very hard to achieve, especially with more nodes. Because of the decrease in overall computation time with more nodes, but increased communication. Therefore, the irony is that the faster and more computation nodes there are, the more the communication time will exceed the computation time. The training becomes communication bound, hindering the scaling to more than a few nodes. The limited bandwidth limits the use of a lot of network traffic, which is needed since for a large model, there are a lot of weights and gradients that need to be communicated. One of the possible solutions analyzed by Keuper et al [30] is the increase of the minibatch sizes. However, they come to the same conclusion as Keskar et al [29] that the increase of the minibatch size decreases the validation accuracy of the model. Therefore, they propose to try to reduce the model and data size, like Seide et al [28] already argued.

5.2 Improvement

An example of a paper that tries to improve communication through model reduction is the paper by Strom [31]. The solution, according to them, is to change the rate at which a change of an individual weight is communicated with other nodes. Normally, all the weights would be communicated with other nodes equally as much at the same times. But by limiting the number of weights to be communicated, one can reduce the amount of communication needed. Strom [31] claims that his method can reduce the communication needed by three orders of magnitude. Since the weight updates are of the same size as the model, it takes a lot of communication bandwidth to synchronize these

weights after each iteration. Strom [31] argues that due to the fact that these sub-gradients are very sparse and are often even sparser than the weight distribution, only a small fraction of these weights are required to be updated after each minibatch. He says that elements of the gradient that are near zero can be delayed longer than the minibatch size. Therefore, he proposes a method that makes use of a threshold to filter which weight updates to communicate with other nodes and which not. This is called gradient sparsification. Strom [31] also implements a version of gradient quantization, which lowers the amount of bits needed to represent a gradient and hence lowering the amount of data that needs to be send. He has found that 1 bit is enough to represent a gradient. The results are encouraging in that the method allows for scaling up to 80 nodes, compared to just the 3 and 16 found by Seide et al [28] and Keskar et al [29]. Furthermore, his experiments show that the method becomes more efficient, the larger the model is. These experiments were run on the AWS cloud and Strom [31] mentions that the results closest in comparison are from experiments run on HPC clusters with Infiniband networks, but nevertheless, the results from Strom [31] were unique at the time. However, the paper does not show a good comparison with other results, just saying that it is so, and Strom [31] makes the too easy assumption that his solution practically solves the entire distributed deep neural network training problem without any real comparison and further research on different types of infrastructures and different types of problems.

Wen et al [32] propose another solution to the communication problem, called 'TernGrad'. Their method is a version of gradient quantization. 'TernGrad' uses ternary gradients, which requires only three levels: -1, 0 and 1. This can greatly reduce the communication time by reducing the overhead caused by gradient synchronization. While proofing the convergence of their method, they use the assumption of a gradient bound and as shown in the paper by Jiang et al [33], this assumption might not be a realistic one. Through the experiments they confirm that larger batch sizes cause accuracy degradation as proven by Keskar et al [29]. However, Wen et al [32] say that the inherent noise of TernGrad might help large-batch methods to converge to flat minimizers as it does with the small-batch methods. Their experiments show that TernGrad improves the accuracy of large-batch methods, which is an asset of TernGrad since often large-batch sizes are required due to their faster training times.

The two methods by Strom [31] and Wen et al [32] are analyzed by Jiang et al [33], where the reason for the paper's existence is the lack of theoretical proof concerning the convergence of the methods. In previous literature, the methods of gradient sparsification were mainly based on heuristics and hence there was no guarantee of convergence. According to Jiang et al [33] there was no real understanding on how gradient sparsification and quantization affects the convergence rate. Their conclusion is that the asymptotic convergence rate of full-precision SGD (O(1/sqrt(MK))) can be achieved if the hyperparameters have the right configuration. This rate implies linear speedup across multiple machines. When analyzing the method proposed by Strom [31], they come to the conclusion that the threshold does not directly affect the convergence rate. Instead, a larger threshold means fewer communication through exchanges of gradients, which increases the variance of the parameters in the different nodes. This leads to a slower convergence rate. Therefore, it is important to change the threshold adaptively to keep the variance as low as possible. When analyzing TernGrad [10], they come to the conclusion that, in theory, generally TernGrad does not achieve the linear convergence rate. However, it can be achieved when the gradient components are more evenly distributed, since this means that there are fewer quantization levels needed and the three from TernGrad will suffice.

Finally, up to this point, most papers and research have been focused on data parallelism and model parallelism to parallelize deep neural networks. Jia et al [34] offer an alternative called 'FlexFlow' that automatically searches for the most efficient way to parallelize deep neural networks on a certain machine. It does this by using a randomized guided search (Markov Chain Monte Carlo) through the SOAP space. Soap stands for sample, operation, attribute and parameter. Operations talks about the way different operations in a deep neural network are parallelized. The sample and parameter dimensions describe how the training data and the model parameters are distributed over the different nodes. The attribute dimension describes the way that different attributes in a sample are divided. Since the SOAP space is very large, a challenge was to efficiently search through the space. For that purpose, FlexFlow implements an execution simulator that is much faster than real executions. This can then be used to estimate the execution times. FlexFlow changes the parallelization optimization problem into a cost minimizing problem and heuristically searches through the space for the best strategy. The strategies discovered by FlexFlow reduce the overall communication cost and the overall task computation time, according to Jia et al [34]. However, they do not thoroughly specify

how they came to this conclusion, so this statement should be further investigated. But if it is true, it shows great promise for the optimization of parallelizing deep neural networks over a lot of nodes.

To conclude this section, at the moment the research on how to best distribute deep neural networks either using data parallelism and model parallelism is still in need of improvement. Although solutions have been proposed to solve the communication bottleneck of data parallel SGD, these do not fully solve the problem yet or need more proof, meaning that at the moment it does not scale well yet to a lot of nodes. Hence, further research is needed to explore different ways to optimize the communication, to look at different ways to parallelize SGD or to investigate and design a different training algorithm for DNNs that is better parallelizable.

6 Discussion and Conclusion

Deep neural networks have brought a new wave of machine learning, and have been widely valued from academia to industry. With the increase of training data sets and the increasing complexity of the network scale, the training time of deep neural networks has become longer and longer. Therefore, the parallelization of deep neural networks is an important basis for accelerating model training. However, due to the complex process, the number of iterations, and the high computational complexity, there are some challenges and bottlenecks in the parallelization of deep neural network as seen in section 5. In this paper, the current deep neural network parallelization technologies were summarized. The methods for distributing DNNs on the cloud and the performances of DDNNs have been analyzed. Below we propose some worthy explorations for future research, based on the detected challenges:

(1) Performance portability of parallel deep neural network algorithm

Due to the diversity of development languages, the parallel deep neural network algorithm developed for a heterogeneous computing hardware must invest a lot of human resources for code rewriting and performance optimization when running on other parallel computing hardware. One of the current solutions is to use a cross-platform programming language, like OpenCL. However, due to the huge differences in the internal structure of heterogeneous computing hardware, the code written by OpenCL can not achieve performance portability[35]. The performance of the model when running on some heterogeneous computing hardware is quite different from the peak theoretical calculation of the hardware. Therefore, the performance portability of parallel deep neural networks urgently needs to be solved by researchers.

(2) Automatic division of tasks in model parallelism

The existing research results show that the parallelization of the deep neural network model mainly aims at the designed neural network structure by manually dividing the network and mapping it to different computing devices. Manually dividing the network may result in an unbalanced load on the computing nodes due to the inaccurate estimation of the running time of the task load and to a less efficient parallelization strategy[36]. To realize the automatic division of the network model and achieve load balancing, it also faces the problem of how to construct an accurate task scheduling algorithm. FlexFlow [34] is a recent implementation of this and shows great promise, but further research is needed.

(3) Challenges faced by data parallelism

For the future development of data parallelism, we can proceed in two directions. The first is to design a distributed stochastic gradient descent algorithm with fast convergence and low communication cost from the perspective of the algorithm[37]. In other words, try to change the inherent sequential nature of the algorithm. The second is to solve the communication bottleneck between different nodes in the cluster[38] as this has not been fully achieved yet.

Finally, it is foreseeable that with the rapid development of heterogeneous computing platforms and the continuous solution of parallelization technical problems, the long training time of neural network applications will be solved and hence the path will be paved for DNNs to be successfully applied to more fields.

References

[1] Liu W, Wang Z, Liu X, et al. A survey of deep neural network architectures and their applications[J]. Neurocomputing, 2017, 234: 11-26.

[2] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016.

[3] Garland M, Le Grand S, Nickolls J, et al. Parallel computing experiences with CUDA[J]. IEEE micro, 2008, 28(4): 13-27.

[4] Diaz J, Munoz-Caro C, Nino A. A survey of parallel programming models and tools in the multi and many-core era[J]. IEEE Transactions on parallel and distributed systems, 2012, 23(8): 1369-1386.

[5] Chapman B, Jost G, Van Der Pas R. Using OpenMP: portable shared memory parallel programming[M]. MIT press, 2008.

[6] Iwainsky C, Shudler S, Calotoiu A, et al. How many threads will be too many? On the scalability of OpenMP implementations[C]//European Conference on Parallel Processing. Springer, Berlin, Heidelberg, 2015: 451-463.

[7] Gropp W, Thakur R, Lusk E. Using MPI-2: Advanced features of the message passing interface[M]. MIT press, 1999.

[8] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[J]. HotCloud, 2010, 10(10-10): 95.

[9] Nguyen G, Dlugolinsky S, Bobák M, et al. Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey[J]. Artificial Intelligence Review, 2019, 52(1): 77-124.

[10] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]//Proceedings of the 22nd ACM international conference on Multimedia. 2014: 675-678.

[11] Abadi M, Barham P, Chen J, et al. Tensorflow: A system for large-scale machine learning[C]//12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016: 265-283.

[12] Chen T, Li M, Li Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv:1512.01274, 2015.

[13] Seide F, Agarwal A. CNTK: Microsoft's open-source deep-learning toolkit[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016: 2135-2135

[14] Collobert R, Bengio S, Mariéthoz J. Torch: a modular machine learning software library[R]. Idiap, 2002.

[15] Bastien F, Lamblin P, Pascanu R, et al. Theano: new features and speed improvements[J]. arXiv preprint arXiv:1211.5590, 2012.

[16] Tang J, Sun D, Liu S, et al. Enabling deep learning on IoT devices[J]. Computer, 2017, 50(10): 92-96.

[17] A. Floarea and V. Sgârciu, "Smart refrigerator: A next generation refrigerator connected to the IoT," 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiesti, 2016, pp. 1-6, doi: 10.1109/ECAI.2016.7861170.

[18] Hsu, Ke-Jou & Bhardwaj, Ketan & Gavrilovska, Ada. (2019). Couper: DNN model slicing for visual analytics containers at the edge. 179-194. 10.1145/3318216.3363309.

[19] Gangrade, Aditya & Acar, Durmus & Saligrama, Venkatesh. (2020). Budget Learning via Bracketing.

[20] Zhou, Li et al. "Distributing Deep Neural Networks with Containerized Partitions at the Edge." HotEdge (2019).

[21] S. Teerapittayanon, B. McDanel and H. T. Kung, "Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 328-339, doi: 10.1109/ICDCS.2017.226.

[22] Lee, In & Lee, Kyoochun. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons. 58. 10.1016/j.bushor.2015.03.008.

[23] Iandola, Forrest & Moskewicz, Matthew & Ashraf, Khalid & Keutzer, Kurt. (2016). FireCaffe: Near-Linear Acceleration of Deep Neural Network Training on Compute Clusters. 2592-2600. 10.1109/CVPR.2016.284.

[24] Shazeer, Noam & Cheng, Youlong & Parmar, Niki & Tran, Dustin & Vaswani, Ashish & Koanantakool, Penporn & Hawkins, Peter & Lee, HyoukJoong & Hong, Mingsheng & Young, Cliff & Sepassi, Ryan & Hechtman, Blake. (2018). Mesh-TensorFlow: Deep Learning for Supercomputers.

[25] Bonomi, Flavio & Milito, Rodolfo. (2012). Fog Computing and its Role in the Internet of Things. Proceedings of the MCC workshop on Mobile Cloud Computing. 10.1145/2342509.2342513.

[26] Dean, Jeffrey & Corrado, G.s & Monga, Rajat & Chen, Kai & Devin, Matthieu & Le, Quoc & Mao, Mark & Ranzato, Aurelio & Senior, Andrew & Tucker, Paul & Yang, Ke & Ng, Andrew. (2012). Large Scale Distributed Deep Networks. Advances in neural information processing systems.

[27] Abadi, Martin & Agarwal, Ashish & Barham, Paul & Brevdo, Eugene & Chen, Zhifeng & Citro, Craig & Corrado, G.s & Davis, Andy & Dean, Jeffrey & Devin, Matthieu & Ghemawat, Sanjay & Goodfellow, Ian & Harp, Andrew & Irving, Geoffrey & Isard, Michael & Jia, Yangqing & Kaiser, Lukasz & Kudlur, Manjunath & Levenberg, Josh & Zheng, Xiaoqiang. (2015). TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[28] Seide, Frank, et al. On parallelizability of stochastic gradient descent for speech DNNs. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014. p. 235-239.

[29] Keskar, Nitish Shirish, et al. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836, 2016.

[30] Keuper, Janis; Preundt, Franz-Josef. Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability. In: 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC). IEEE, 2016. p. 19-26.

[31] Strom, Nikko. Scalable distributed DNN training using commodity GPU cloud computing. In: Sixteenth Annual Conference of the International Speech Communication Association. 2015.

[32] Wen, Wei, et al. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In: Advances in neural information processing systems. 2017. p. 1509-1519.

[33] Jiang, Peng; Agrawal, Gagan. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In: Advances in Neural Information Processing Systems. 2018. p. 2525-2536.

[34] Jia, Zhihao; Zaharia, Matei; Aiken, Alex. Beyond data and model parallelism for deep neural networks. arXiv preprint arXiv:1807.05358, 2018.

[35] Zhang Y, Sinclair M, Chien A A. Improving performance portability in OpenCL programs[C]//International Supercomputing Conference. Springer, Berlin, Heidelberg, 2013: 136-150.

[36] Mirhoseini A, Pham H, Le Q V, et al. Device placement optimization with reinforcement learning[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 2430-2439.

[37] Xing E P, Ho Q, Xie P, et al. Strategies and principles of distributed machine learning on big data[J]. Engineering, 2016, 2(2): 179-195.

[38] Seide F, Fu H, Droppo J, et al. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns[C]//Fifteenth Annual Conference of the International Speech Communication Association. 2014.

7 Contribution

Table 3 shows the contribution of each team member. First, we tried to understand and analyze the expected result and writing tasks before diving into reading. After reading a few papers, we came up with the thesis and structure of the report and divided the tasks according to this structure. Specifically, Dongqi was mainly responsible for writing Deep Learning and Parallel programming and neural network framework. The sections Abstract, Introduction, Discussion and Conclusions were drafted by Dongqi, and everyone participated in the revision. DNN on the cloud and edge was completed by Juan, and Carlijn was responsible for the performance analysis writing.

Table 3: Group member contribution

Task Description	Member
Abstract, Introduction	All
Deep Learning	Dongqi PU
Parallel programming and neural network framework	Dongqi PU
DNN on the cloud and edge	Juan Agustin Tibaldo
Performance Analysis	Carlijn Nijhuis
Discussion and Conclusion	All

Throughout the literature study process, we held many online meetings. Although the impact of the Coronavirus outbreak prevented us from having face-to-face conversations, network tools, like Discord, helped us to connect. These meeting were to determine the research direction, because the original topic was somewhat general, to find a more suitable and narrow topic, to find suitable literature to support our study and research and discuss our findings. To this end, we searched and browsed a large number of papers, and extracted their keywords and research topics. These were then categorized to get a global overview of the research about DDNNs.

Serverless Computing and Function as a Service

Glenn Bond
Faculty of Science
University of Amsterdam
Amsterdam, Science Park 904
glennbond39@gmail.com

Corneliu Soficu
Faculty of Science
University of Amsterdam
Amsterdam, Science Park 904
Corneliu.Soficu@gmail.com

Philip Roeleveld
Faculty of Science
University of Amsterdam
Amsterdam, Science Park 904
philiproeleveld@gmail.com

Abstract

Based on its advantages, will FaaS become the future in the field of serverless computing? This paper aims to answer that question by means of a literature study. It contains topics such as the history of Cloud Computing and how FaaS and Serverless Computing in general originated. It describes how a FaaS architecture works. It also identifies the largest providers that offer FaaS and gives examples of Serverless Platforms and Serverless Frameworks. Furthermore, some current research trends and topics in the field of FaaS are examined, as well as what problems are currently faced in the field of FaaS. Additionally a review is done of some performance results relating to FaaS. And finally a look is cast at the future to discuss how serverless computing and FaaS in paricular will end up.

KEYWORDS

Serverless Computing, Cloud Computing, Service, FaaS

1 Introduction

Nowadays there is more and more development in the field of cloud. From the development of new platforms to the development of new cloud frameworks. Among these is the interesting evolution of serverless computing and Function as a Service (FaaS). There has been a lot of interest for this subject recently, from the scientific community as well as industry leaders, and we have chosen it as the topic for the literature study.

In order to structure our study, we pose one main question followed by multiple sub-questions. The main question is: On the basis of its advantages, will FaaS become the future in the field of serverless computing?

This main question is divided into three sub-questions. The first sub-question is: What are the pros and cons of FaaS? The second sub-question is: Who are the major providers of serverless platforms? And the third sub-question is: What are the advantages of migrating a cloud infrastructure to serverless?

These questions are divided into different subsections. The first section 2 explains how FaaS originated. What history has taken place in the field of cloud computing such as the rise of IaaS,

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

PaaS and CaaS and the following serverless computing technologies such as FaaS and BaaS. Section 3 explains how FaaS works technically and what providers offer FaaS in industry. A number of examples of serverless platforms and frameworks are also mentioned. In 4 will be explained what types of research trends and topics there are regarding serverless at the moment. In 5 the current problems regarding serverless are discussed. Section 6 zooms in on performance results about serverless computing. The last section in 7 describes how we see the future of serverless computing. Finally, the literature study ends with a discussion and a conclusion.

2 Cloud/Serverless Computing and Function as a Service

In order to describe FaaS, we will first explain what history has taken place in the field of cloud computing to arrive at the serverless paradigm and explain how FaaS relates to other Cloud Computing services.

In 2006 a new development took place in the field of application hosting. In that year, Amazon's subsidiary Amazon Web Services (AWS) developed EC2 (Elastic Compute Cloud). EC2 is seen as one of the first Infrastructure as a Service (IaaS) platforms. This made it possible to rent out compute capacity and run server applications over the internet. This development brought many advantages. For example, the costs were reduced for users because all of the server management work was handled by the service provider. Risk factors were reduced because when a hardware failure occurred, a new machine could easily be requested. In this case, the user only had to reinstall the application. Infrastructure costs were reduced due to the flexibility offered by running hosts for a short period of time. Another advantage was the flexibility offered by scaling servers up and down. For example, it was not necessary to purchase a large number of servers for a short period of time. Another advantage is that when a new application needs to be tested it used to take a long time to get the server ready, but with EC2 this could be done within a much shorter time (1).

After IaaS came Platform as a Service (PaaS) in which the user only has to take the applications into account. PaaS layers on top of IaaS. In this case, the provider is responsible for everything else to keep the platform up and running (1).

Due to recent developments it is now possible to offer containers developed in PaaS work as services. For this purpose, the cloud service model CaaS can be applied. CaaS creates the opportunity to make applications independent of the specific underlying platform. The application is disconnected from that specific environment so that it can be run anywhere. Besides IaaS, PaaS and CaaS are therefore also seen as techniques for infrastructural outsourcing by outsourcing even more technologies to others (2).

The big advantage of serverless computing is that it has the same advantages as mentioned before. Serverless consists of two areas with different technologies, these are Back-end as a Service (BaaS) and Functions as a Service (FaaS) (1).

BaaS is an approach to connect web and mobile applications to back-end cloud storage and processing. In addition, BaaS also offers common features so that manual programming is no longer necessary, e.g. social networking integration. BaaS provides an infrastructure that automatically optimizes with resources such as data and API driven services. In this way, the development of the back-end can be accelerated (3). Because BaaS is more focused on replacing components it could also be classified as Software as a Service (SaaS).

In addition to BaaS, serverless also includes FaaS. FaaS is a serverless solution that services individual function executions and obscures the fact that the functions are hosted on servers. FaaS is therefore seen as an independent service (4).

FaaS allows developers to provide code that runs in an isolated environment for every execution. In doing so, FaaS offers vast scalability, including all the way down to zero, and associated pricing in a pay-as-you-go model. This scaling is performed on the basis of the incoming events. With FaaS, separate functions become part of a large application. And a big advantage is that FaaS functions can be quickly activated and executed on demand (17). We therefore use the following definition of FaaS: "Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests" (6).

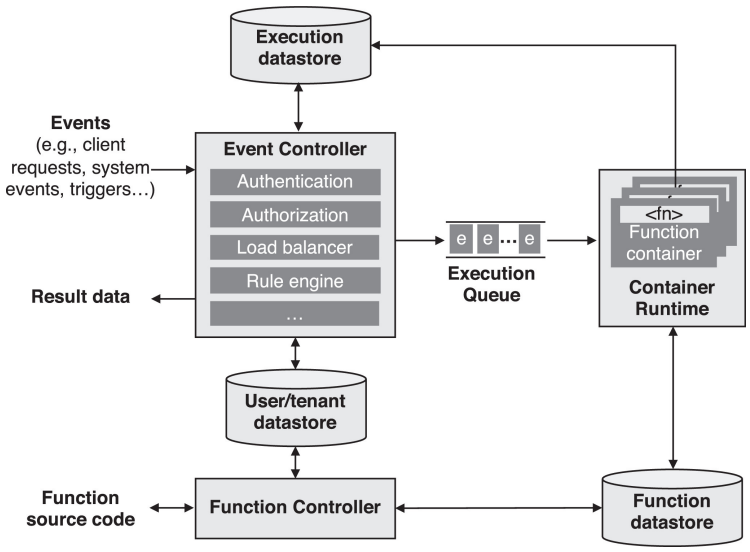


Figure 1: FaaS system architecture. From Leitner et al. (17).

3 How does FaaS work?

Here we will discuss the technical details of FaaS. We describe some providers that offer FaaS. However, this is limited to only the four largest companies. Next, a number of examples are given about serverless platforms and serverless frameworks.

First of all, we provide an explanation of the system architecture of FaaS. To this extent, we refer to the diagram in figure 1.

Functions are created, updated and deleted by users through the use of a function controller. This is how event triggers, rules and function source code are developed and deployed. The source code invoked by the functions is uploaded to a function datastore. Triggers and rules are stored separately in a user/tenant datastore.

The process of function execution starts as follows. Incoming events are processed by the event controller. The event controller determines when to execute which function based on the triggers and rule configured in the user/tenant datastore. The event controller generally also handles load balancing and determines which container runtime (or host) will execute each function. Then the execution will be queued.

The container runtime in its turn picks up executions from the queue. And each functio is executed in an isolated container. The execution results are stoerd in the execution datastore and possibly used for additional functions. Finally, the results are brought back to the clients by the event controller (17).

3.1 Evolution to Serverless

This section will present the underlying technologies and their evolution towards the current paradigm of serverless. Virtualization abstracts away the physical machine and allows the same physical resources to be shared between multiple applications and users. With the emergence of the Internet, virtualization became widespread and it started being used for shared hosting through virtual private servers. The development of cloud computing also allowed the possibility of making virtual resources available over the internet. Digital containers were created to offer protection of their content from external abuse, in a similar fashion to virtual machines. Extending on the works of Linux Containers (LXC), Docker became a system for convenient container orchestration by offering an ecosystem based on digital containers. Serverless computing is the later iteration of the long-term process of virtualization abstractions, by exposing to the user abstract resources (Ex: Functions) that are mapped to concrete resources such as containers (11).

Code as functions is the central point in the serverless paradigm and it evolved from previous technologies such as the IBM's Customer Information System (CICS), RPC, stored procedures for databases or Common Gateway Interface (CGI) for web servers which aimed to bring support for executing functions to specific domains. Serverless computing aims to provide a full abstraction for event-driven execution of generic functions, in contrast to these previously described context-specific implementations (11).

Naming and discovery of services are essential when managing a large cloud infrastructure. Current approaches for service naming and discovery are based on the Lightweight Directory Access Protocol (LDAP) and URI. LDAP is used for enabling distributed directory services over TCP/IP. URI has the role of assigning unique identifiers encoded as character strings to resources. The serverless paradigm adopts service naming and discovery and extends it with function versioning and aliases. By using versioning, the developer can use different versions of a function simultaneously and by using aliases as mutable pointers to a version, a transition of a version between one stage to another can be done without changing the deployed application (e.g., from development to production) (11).

The concept of functions as computation is also essential to serverless computing. This concept stems from ever-higher-level abstractions that started with functional programming that departed from procedural programs where the developer was allowed to manage abstract data types and control flows instead of controlling specific details about memory and processors. With time, we evolved to interconnecting services using service-oriented-architecture (SOA) based on REST. These developments ultimately led to microservices that provide specific functions within self-contained applications. Considering this, serverless development represents an evolution that is based on the aforementioned concepts and can be characterised as an hyper-specialization of services (11).

Over the past decades, we have moved to a different form of expressing concurrency, by using a declarative form where workflows describe the structure of applications and where concrete execution and synchronization of workflow tasks are left for the runtime system. This model of describing concurrency using a declarative form has many applications and is successfully applied to serverless computing. As an example providers such as AWS or Microsoft Azure allow for an easy setup of concurrency using configuration files. (11).

Event-driven programming represents a basis on which serverless works and has evolved from the traditional way of developing programs in a synchronous way. With the extended use of high-level languages and advanced operating systems as well as with the rise in internet usage, event-driven distributed systems became widely used. Serverless computing leverages this idea by using well-defined protocols and ways to manage events such as adopting message queues (11).

3.2 Providers

There are many providers that deliver serverless architecture. The largest companies are Amazon, IBM, Google, and Microsoft.

Amazon offers AWS Lambda. Lambda is one of the largest players in the field of FaaS and is therefore seen as the meaning of serverless. AWS Lambda offers the largest range of services of any other provider.

Azure Functions is almost the same service as Amazon. Azure Functions offers about the same services like Amazon. However, Azure is more focused on Microsoft services.

Google Cloud Functions is the cloud provider of Google. In the early years around 2017, Google Cloud was the variant that lagged behind the competition. However, Google later managed to keep up with the competition.

IBM Cloud Functions is one of the newer providers. IBM Cloud Functions offers OpenWhisk and Apache OpenWhisk. IBM Cloud Functions offers OpenWhisk as a managed infrastructure within their cloud services. For the use of open-source solutions, the user has the choice to use Apache OpenWhisk (5).

3.3 Serverless Platforms and Serverless Frameworks

Nowadays there are several serverless platforms and serverless computing frameworks in use. Examples of open-source serverless computing platforms are OpenLambda and Galactic Fog.

OpenLambda is a reference architecture for serverless platforms (9). It is an open-source implementation of the FaaS paradigm. It uses independent modules that communicate with each other through APIs. (19) These subsystems manage function execution by coordinating them and distributing them among available hosts. So these subsystems include execution engine, load balancer and database (7).

Galactic Fog Gestalt Laser also known as Lambda Application Server is Galactic Fog's serverless solution. This platform was launched including a CaaS abstraction layer and an enterprise integration layer. Gestalt is a high-performance serverless engine that supports the most common programming languages (18). The framework includes functions such as policy management and security capabilities.

Examples of open-source serverless computing frameworks are Kubeless and OpenFaaS. Kubeless is based on the Kubernetes API, using its Custom Resource Definitions (CRDs). These CRDs are used to create functions as custom objects in Kubernetes. In this way, native Kubernetes APIs can interact with functions. The Kubeless controller scans for changes to the function objects and removes resources when a function object is done executing (8).

OpenFaaS is another open-source serverless framework based on Docker and Kubernetes. The OpenFaaS CLI is used to develop functions and deploy them on the platform. The CLI ensures that the function is packaged into a Docker container. This container then exposes an entrypoint such that the function can be executed within the framework. And an API gateway is used to access these functions (8).

4 Current research trends and research topics

On the basis of literature research, various trends and research topics were investigated. We have included the most recent research areas in the paper. This has led to certain research possibilities, such as how to handle SLA's (service-level agreements) and using legacy code in serverless.

One of the research possibilities lies in the SLA area, about laying down agreements between the provider and the customer. This includes the performance and availability of the application. However, this is quite difficult to document in the area of serverless computing. This is because the application depends on an ecosystem of services that are beyond the control of the serverless platform, making it difficult to provide QoS (Quality of Service) guarantees (6). As a result, enforcement needs to be applied inbetween the functions and APIs. This will require measurements to be made of these types of services. For example, a third party can be called in, whereby an evaluation system can be used to examine the bottlenecks.

There are also research opportunities to investigate how to convert existing legacy code into smaller components and make modifications to fit the serverless framework. This is usually non-trivial because serverless applications need to be designed and constructed differently from most legacy applications (6). Research opportunities include identifying ways to prevent starting from scratch and wasting the time and money spent on development of the legacy code in the process.

5 Current Problems

The serverless paradigm has many advantages and it benefits both consumers and providers but there are also some disadvantages that must be mentioned. One such drawback from the perspective of the consumer is that the FaaS model that is offered by the platform might be too constraining for some applications. As an example, when consumers might want to use the platform, they might discover that various libraries might not be available or that certain dependencies might not be up to date with the latest version (9).

From the perspective of the provider offering FaaS capabilities, it has to ensure good reliability of the product which means managing issues related to the user's functions life cycle. This implies that providers must take care of scalability or fault tolerance issues in an application-agnostic manner. Considering this, it becomes necessary for developers to have a great understanding of the behaviour of the platform so that they can design applications around these capabilities (9).

Another drawback related to FaaS revolves around the risk of vendor lock-in. This issue stems from the fact that providers tend to offer an entire ecosystem that can be used in conjunction with the

user's functions. Such auxiliary services that augment the functions are: services for managing the state, recording and monitoring logs, sending alerts, triggering events or performing authentication or authorization. These auxiliary services can be attractive to developers and represent an alternative source of revenue for the cloud provider (9).

We can also argue that FaaS is not very compatible with Distributed Computing. This is because of the lack of network addressability of serverless functions. In order for two functions to work together, they have to transfer data using slow and expensive storage. This stymies basic distributed computing as this field relies on protocols executing fine-grained communication between nodes. Such communication includes leader election, membership, data consistency or transaction commits (10).

6 Performance

The goal of this section is not to perform a rigorous quantitative literature study as this would involve exhaustively finding all reliable sources about the performance results we are interested in. Instead, this section only aims to provide context and qualitative review about some of the performance research done about serverless computing.

It should be noted that many studies draw a comparison between serverless computing and other cloud solutions, but this is usually a comparison about cost instead of performance and thus left open.

Most studies regarding the performance of FaaS solutions consider at least AWS Lambda and Azure Functions, but many also include Google Cloud Functions and IBM OpenWhisk. Out of these four, Azure Functions immediately stands out as the only provider based on Windows rather than Linux, and also the only provider to use dynamic memory allocation rather than user-defined.

To evaluate performance the first step is to define metrics. McGrath and Brenner propose two such metrics to address elasticity and infrastructure retention/continuous deployment being concurrent execution performance and cold start latency respectively (12). These metrics were adopted by Lloyd et al. and expanded upon to consider the effects of load balancing, provisioning variation, and memory reservations (13). Lee et al. also adopt the elasticity metric but forego the infrastructure retention metric in favour of more traditional metrics such as CPU, disk, and networking throughput (14).

One well-known variable in measuring the performance of computer systems, in general, is the distinction between cold and warm starts. This is especially important for FaaS because cold starts are an important problem faced by serverless computing. Lloyd et al. go as far as to warn the reader that "the significance of container initialization overhead cannot be overlooked!", which is also supported by their findings including a performance degradation of up to 15x between a warm and cold start (13). See for instance figure 2 illustrating the performance drop brought about by a cold start, and figure 3 indicating that such a performance drop can become a reality as soon as after 40 minutes of idle time. Other studies rather divert the attention from this problem of cold execution by controlling for it in their methodologies, usually performing their analyses exclusively with warm executions.

Another approach is to use micro-benchmarking to compare FaaS solutions. Malawski et al. opt for CPU-intensive tasks to benchmark the solutions and also demonstrate the difficulty in measuring performance caused by the heterogeneous nature of FaaS (15). They show a strong correlation between memory reservation and CPU performance in the case of AWS Lambda and Google Cloud Functions, but not at all for Azure Functions and Apache OpenWhisk. Figure 4 illustrates what this correlation looks like in the case of AWS Lambda. Back and Andrikopoulos use well known algorithmic tasks of varying loads for their microbenchmark (16). Unfortunately, they do not control for the cold start phenomenon and note a small number of data points as a threat to validity. Despite this, they do reach the same conclusion about the correlation between memory reservation and performance.

In general, much work is still to be done for serverless computing when it comes to performance. Most notably the question which provider to go with for a given use case is largely left unanswered by the research done so far.

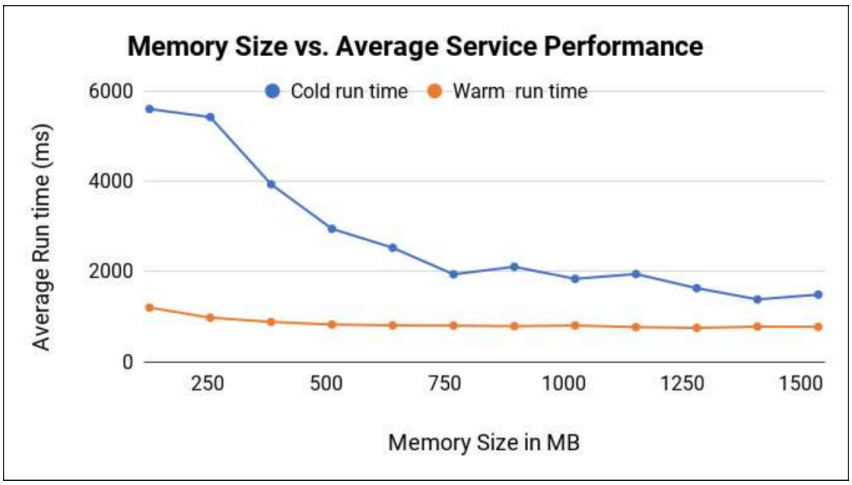


Figure 2: The difference between cold and warm execution time for diferent memory sizes on AWS Lambda. From Lloyd et al. (13).

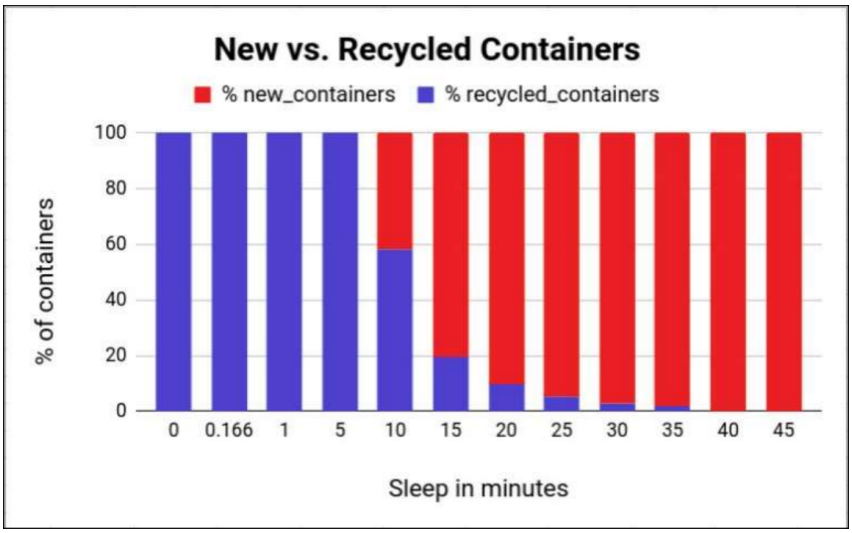


Figure 3: Container retention percentage for different idle times on AWS Lambda, reaching 100% new containers after 40 minutes. From Lloyd et al. (13).

7 Future Trends

Based on the problems currently faced by serverless, we can also look towards the future. First of all, as highlighted by multiple sources (9)(17), we will have to see more and better tooling built around the concept. Without this FaaS is just not worth the hassle for many businesses if the alternative coarser cloud solutions such as IaaS do have a lot of support. However, because of the appeal of pay-as-you-go as well as the interest already demonstrated today, we do think such tooling will be developed and this hurdle will largely be overcome.

As FaaS matures, we should see an ever-growing list of programming languages and libraries supported on the major platforms, especially if the demand for serverless increases. Thus problems such as language support, missing libraries and outdated dependencies should be resolved for most users. To address the issue of vendor lock-in we might see some unifying standard start to get a foothold, fulfilling a similar role as Docker has taken in the wider cloud ecosystem.

Some other problems on the other hand are less solvable because they are artifacts of the nature of serverless. The most prominent example of this is the latency issue. While it is conceivable

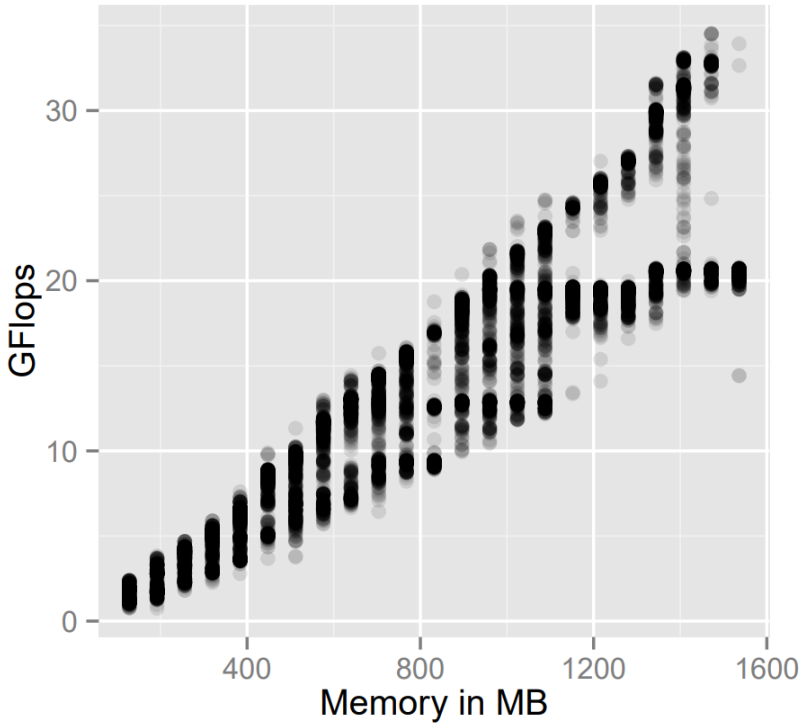


Figure 4: GFLOPS vs. memory for a Linpack benchmark on AWS Lambda, showing a clear correlation between the two. From Malawski et al (15).

that this issue will be diminished significantly, the best-case scenario is still worse than that of the non-serverless alternative. For that reason we do not believe serverless will "take over". Rather we think it will find it's a niche in the use cases it excels at. For established services that means cases with bursty or predominantly idle workloads where elasticity is paramount, and ideally those that lend themselves to a stateless approach (9). Additionally FaaS is a perfect candidate for prototyping new ideas, because of the relatively low cost and ease of deployment at a smaller scale. This is also an attractive prospect for starting businesses. However despite the promise of scalability we think it is likely that we will see such businesses migrate their services as their needs grow because the cost of FaaS could start to overshadow that of other possible solutions.

Discussion and Conclusions

We have given an overview of the state of serverless computing, and Functions-as-a-Service in particular, as well as the research efforts that have been dedicated to it. A history has been sketched that lead to the onset of serverless from two perspectives. On the one hand, we have recollected the many precursory as-a-Service proposals that predate FaaS. On the other hand, we have looked at the many technologies that lead to serverless becoming possible from a technical point of view.

Today, many cloud providers offer a FaaS solution for serverless computing. And furthermore, there have been some open source initiatives as well such as OpenLambda and OpenFaaS. The increasing popularity of FaaS, in particular, has been accompanied by interest from the scientific community as well. On-going research includes such issues as how to manage service-level agreements, and how to migrate legacy code to this mostly stateless environment in the industry.

The problems currently faced by FaaS are multiple, but not insurmountable. Vendor lock-in and language support, in particular, are problems that can be overcome in the future. Other problems are less easily addressed. One of these is the lack of network addressability that hinders serverless distributed computing. Another is the performance drop after cold starts, which is exacerbated when infrastructure retention is low. These problems, however, are more indicative of the limitations of FaaS, and serverless as a whole. Thus the conclusion remains at least for now that serverless has

many applications, but is not a silver bullet solution to be applied to any and all use cases. However, we would like to be proven wrong in this regard. Thus we would welcome any increased interest in FaaS and serverless computing.

References

[1] Roberts, M., & Chapin, J. (2017). *What Is Serverless?*. O'Reilly Media, Incorporated.

[2] Hussein, M. K., Mousa, M. H., & Alqarni, M. A. (2019). *A placement architecture for a container as a service (CaaS) in a cloud environment*. Journal of Cloud Computing, 8(1), 7.

[3] Lane, K. (2015). *Overview of the backend as a service (BaaS) space*. API Evangelist.

[4] Fox, G. C., Ishakian, V., Muthusamy, V., & Slominski, A. (2017). *Status of serverless computing and function-as-a-service (faas) in industry and research*. arXiv preprint arXiv:1708.08028.

[5] Lobastov, I. (2019, 11 maart). *Comparing Serverless Architecture Providers: AWS, Azure, Google, IBM, and Other FaaS Vendors - DZone Cloud*. Retrieved from https://dzone.com/articles/comparing-serverless-architecture-providers-aws-az

[6] Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). *The rise of serverless computing*. Communications of the ACM, 62(12), 44-54.

[7] Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2016). *Serverless computation with OpenLambda* In 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16).

[8] Mohanty, S. K., Premsankar, G., & Di Francesco, M. (2018, December). *An Evaluation of Open Source Serverless Computing Frameworks*. In CloudCom (pp. 115-120).

[9] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., ... & Suter, P. (2017). *Serverless computing: Current trends and open problems*. In Research Advances in Cloud Computing (pp. 1-20). Springer, Singapore.

[10] Hellerstein, J. M., Faleiro, J., Gonzalez, J. E., Schleier-Smith, J., Sreekanti, V., Tumanov, A., & Wu, C. (2018). *Serverless computing: One step forward, two steps back*. arXiv preprint arXiv:1812.03651.

[11] Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uță, A., & Iosup, A. (2018). *Serverless is more: From PaaS to present cloud computing*. IEEE Internet Computing, 22(5), 8-17.

[12] McGrath, G., & Brenner, P. R. (2017, June). *Serverless computing: Design, implementation, and performance*. In 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW) (pp. 405-410).

[13] Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L., & Pallickara, S. (2018, April). *Serverless computing: An investigation of factors influencing microservice performance*. In 2018 IEEE International Conference on Cloud Engineering (IC2E) (pp. 159-169).

[14] Lee, H., Satyam, K., & Fox, G. (2018, July). *Evaluation of production serverless computing environments*. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 442-450).

[15] Malawski, M., Figiela, K., Gajek, A., & Zima, A. (2017, August). *Benchmarking heterogeneous cloud functions*. In European Conference on Parallel Processing (pp. 415-426). Springer, Cham.

[16] Back, T., & Andrikopoulos, V. (2018, September). *Using a microbenchmark to compare function as a service solutions*. In European Conference on Service-Oriented and Cloud Computing (pp. 146-160). Springer, Cham.

[17] Leitner, P., Wittern, E., Spillner, J., & Hummer, W. (2019). *A mixed-method empirical study of Function-as-a-Service software development in industrial practice*. Journal of Systems and Software, 149, 340-359.

[18] Lynn, T., Rosati, P., Lejeune, A., & Emeakaroha, V. (2017, December). *A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms.* In 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) (pp. 162-169).

[19] HoseinyFarahabady, M. R., Zomaya, A. Y., & Tari, Z. (2017). *A model predictive controller for managing qos enforcements and microarchitecture-level interferences in a lambda platform.* IEEE Transactions on Parallel and Distributed Systems, 29(7), 1442-1455.

A Contribution per Group Member

The following table contains the information about which member was responsible for each section. It should also be noted that both the collection of sources, as well as the reviewing/proofreading effort was a joint effort carried out by all three members.

Section	Responsibility
Introduction	Corneliu
Cloud/Serverless Computing and Function as a Service	Glenn
How does FaaS work?	Glenn
Evolution to Serverless	Corneliu
Providers	Glenn
Serverless Platforms and Serverless Farmeworks	Glenn
Current Research Trends and Research Topics	Glenn
Current problems	Corneliu
Performance	Philip
Future Trends	Philip
Discussion and Conclusions	Philip

Big Data and Cloud

Yuxue Liu
liuyuxue97@gmail.com
Shane Minnema
s.e.minnema@vu.nl
Mansi Mundra
mansimundra.38@gmail.com

June 1, 2020

Abstract

Big data and cloud computing is an ever-growing field in the computer science and technology sectors. This paper focuses on explaining what big data and cloud computing is, and how they relate to each other. We discuss the core principles of big data, common cloud service platforms, and compare the performance of several popular big data/cloud computing frameworks.

1 Introduction

The content of this paper focuses on big data and cloud computing. As two key points of future development in computer field, there are more and more people focus on big data and cloud computing. This paper provides a review for individuals who want to have a brief understanding of those two fields and how they interact with one another. First, we discuss what big data is and go through its five V characteristics. Next, we discuss and compare three common types of cloud computing services, and point out their advantages and disadvantages. Further, we review big data and cloud computing together by introducing and comparing the performance of several popular big data frameworks. Finally, we provide a discussion and conclusion based on our research. For those that do not have experience in the field of big data and cloud computing, sections three and four of the paper may be difficult to understand, but we attempt to explain these as clear as possible. For each section, we also add our own critical analysis and opinions on the topics discussed.

2 Big Data

Conventional data becomes *big data* once it reaches a certain size or complexity to where it becomes problematic or unfeasible to process using traditional

hardware/software setups[1]. In these cases, significantly more time is required if the methods used are not tailored towards processing big data. The act of accessing and storing large amounts of information for data analytical purposes is not a particularly new concept, as popularity of the field began to increase in the early 2000s when Doug Laney first gave his definition big data. His definition still holds relevance today: *"Big data is data that contains greater variety arriving in increasing volumes and with ever-higher velocity"*[2].

Currently, big data plays a role in almost every field and industry in some way. It is a dominant driving force behind the global performance of companies and organizations. In the following subsection we discuss big data in more detail in order to obtain a better understanding of what big data is.

2.1 Three V's of big data

Along with Doug Laney's definition, he also provided the three V's of big data: *Variety*, *Velocity*, and *Volume*. Through these characteristics, we can obtain a better understanding of what big data really is.

2.1.1 Variety

Variety, as the name suggests, refers to different data formats that come from different sources. Big data includes data which is structured, unstructured, and semi-structured. Traditional data has been more formal in the past, and is often from internal data sources. Now, new data is increasingly being sourced externally, mainly using the internet, with most data being completely unstructured. Compared to traditional data, such as personal files and documents, newer data can be in the form of location data, photos, audio, etc.

2.1.2 Velocity

Velocity means data accumulation at high speeds. This is important when considering the massive amount of data produced every day, an estimated 2.5 quintillion bytes of data, and this figure is only going to increase[3]. This year it is forecast each person on the planet will produce approximately $1.7MB$ of data every second. Velocity plays a major role in a big data society compared to others. By speeding up, it saves time and money for companies to collect the data.

2.1.3 Volume

Big data usage indicates the daily creation of enormous volumes of data from various sources, such as social media platforms, enterprise processes, mobile devices, networks, human interactions, etc. For example, Twitter generates an average of approximately 6,000 tweets per second, more than 350,000 tweets a minute, 500 million monthly, and around 200 billion tweets annually.[4]. Big data technologies are well suited, and possibly a requirement, for handling such large quantities of data.

2.2 Two new V's: Value & Veracity

There are two newer additions to the three V's mentioned above, namely Value and Veracity, which were added as key aspects of big data. Value regards the quality of the data that is stored along with its further use. Veracity stems from the idea that the consistency of data is sufficient enough for big data[5].

With the growing development of big data, not only does it bring positive aspects to society, but also raises some questions, such as how truthful the data is. Data flows are unpredictable and may come from various sources apart from the rising velocity and variation of the data. Linking, matching, cleaning, and transforming data across systems can be challenging to achieve for researchers and businesses. Certain industries must also act responsively towards data. An example of this could be a media company reporting on the latest trending news.

2.3 Critical Analysis

The emergence of the big data era has forced many entities to transform and adapt, whether or not they are traditional ones. Ever-increasing amounts of data leads to big data technologies being constantly updated and iterated over. With the continuous improvement in the field, the number of talent is also increasing, and the speed of improving technology is also increasing. It is hard to deny that the era of big data has encouraged many individuals to devote to this promising field and harvest spectacular achievements.

3 Cloud Computing

Cloud computing simply provides the internet, or the cloud, with a faster range of innovation, flexible resources, and economies of scale through computing services[8]. Examples include servers, storage, databases, networking, software, analysis, and intelligence services. The early days of cloud computing was based on simple distributed computing, which helped solve problems with scalability and combining results. With this technology, tens of thousands of data points can be processed in a very short time (i.e. several seconds), so as to achieve powerful network services; however, the advancement of hybrid technologies today, including utility computing, load balancing, parallel computing, network storage, hot backup replication and virtualisation, enable cloud services[9]. Cloud computing is a significant advance. Using cloud computing can help companies save a lot of money in hardware and software that users do not have to purchase.

3.1 Cloud computing services

The extensively adopted and well-established cloud computing services are Infrastructure as a Service(IaaS), Platform as a Service(PaaS), and Software as a Service(SaaS).

3.1.1 Infrastructure as a Service

Infrastructure as a Service is a cloud computing solution that provides central computer components such as virtual servers, storage, and networking devices to an organization or enterprise. IaaS is a hosting method, evolved from traditional hosting methods, that does not need any long term commitment and enables users to provision on-demand resources[10].

Infrastructure as a Service requires wide area connections to the network, routing, and storage facilities. In general, the IaaS provider provides services such as the hardware and administration necessary to store applications and to operate a platform to users using wide area networks. Vendors mostly include facilities such as bandwidth, memory, and storage devices. Additionally, vendors compete with competitive service efficiency and pricing. An enterprise or company does not have to worry about the accommodation, operation, and maintenance of the infrastructure provided as service provider is responsible for these facilities. IaaS can be bought on a pay-as-you-go or contract basis. Most buyers regard the key benefit of IaaS as price stability, as you just have to pay for the resources needed for your applications. There are some examples of IaaS providers such as Amazon Web Services Elastic Compute Cloud (EC2) and Secure Storage Service (S3).

3.1.2 Platform as a Service

Platform as a Service is a cloud computing service owned by a third party in order to support software development by providing software and hardware resources over the internet[11]. Generally, software developers take advantage of this service. The tools and software that developers need to build apps above them can include middle-ware, data base management, operating systems, and development tools. These tools and software services are provided by PaaS service provider over internet to the developer.

Platform as a Service architecture is quite more intricate than architectures of other cloud computing solutions. Completed and in-progress cloud applications, both type of development platforms are offered by PaaS service providers to developers. Mostly PaaS Models consists of the physical infrastructure (i.e. used to buttress software development), software solutions (i.e. tools required to develop applications), and graphical user interfaces. Google AppEngine is an example of PaaS.

3.1.3 Software as a Service

Software as a Service is a model owned by a third party which provides web applications and software access over the internet to the end-users. It can accessed by thin client interfaces such as web browser[12].Software as a Service is a software delivery method which requires good internet connectivity to access the data available online.

SaaS usually is charged on the basis of use (i.e. subscription basis and on the basis of the environment of multiple tenants). A software application to be

used and purchased upon request in Saas model a software provider license. It is more acceptable for business users that they do not need to perform the client installation. Also, users do not require setting up or maintaining the software and its services, as this task is handled by service provider. Some examples of SaaS are Microsoft Office 360, Google docs, AppDynamics, Adobe Creative Cloud, and Google G Suite.[23].

3.2 Pros and Cons of IaaS, PaaS and SaaS

Cloud computing is a topic that covers many concepts regarding internet territory. Understanding the advantages and disadvantages of the various cloud services is quite important in today's world as it helps us to understand how business/organisations, developers, and end users would benefit and/or suffer impediment by opting for cloud computing services[13][14].

3.2.1 Advantages and Disadvantages of IaaS

With the lower infrastructure cost and pay-as-you-go pricing, IaaS is quite *cost effective*[15]. It is an economical option for startups as users do not have to worry about purchasing, maintaining, replacing, or ensuring uptime of any hardware or networking equipment. *On-demand scalability* is one of the major advantages of IaaS. Upgrading of software, hardware, or troubleshoot-equipment problems is done by the IaaS service provider. IaaS help users to scale-up or scale-down according to their requirements. Meanwhile, IaaS provides *operational flexibility* by allowing employees to access the data and files offsite and on-the-go at anytime. As a third party service, IaaS can handle the workload of upgrading and maintaining infrastructure to support business operations. Therefore, it helps businesses to focus on *business growth.* Lastly, IaaS is *highly reliable* as it runs even if hardware components fail, servers go down, or entire data goes offline.

Security is one of the main concern in IaaS environment as users do not have control over their cloud security. Therefore, it might expose sensitive information if system is compromised. *lack of flexibility* is one of the crucial drawbacks of IaaS, as service provider is responsible for upgrading the hardware as well as software and sometimes they do not upgrade the software for some users. This might have major impact on the efficiency of the employees. Additionally, enterprises face some *technical problems* with IaaS, and this can lead to hindering the access of companies to data and applications. Opting for IaaS for a company's infrastructure leads to *over dependency* on the service provider for the company's data. In IaaS, third parties are responsible for *upgrade and maintenance.* This can be drawback if service providers fail to upgrade and maintain their infrastructure regularly, as this might decrease employees efficiency significantly. Lastly, it uses *virtualization services* and also limits *user privacy and customisation*[15].

3.2.2 Advantages and Disadvantages of PaaS

Cost reduction is major advantage of Platform-as-a-service as it not only reduces setup costs, but also reduces installation cost and time due to its already programmed servers loaded on internet[16]. *Control and management* of architecture by user is one of the significant benefits of PaaS as user get full control of developing and deploying software, even though PaaS architecture is provided by the provider. The *software deployment* process is achieved easily and speedily in PaaS as users do not need to create their own hardware/software. PaaS is quite *flexible* as users need to only pay for the features they require. *Less coding* is required with PaaS due to pre-programmed applications and users not having to start coding from the beginning.

Provider or vendor lock-in is one of the major limitation of PaaS[16]. Different providers may have different architecture requirements and they may not use same language, architecture, libraries, APIs, or OS to build and run applications. Therefore, it is quite hard to switch Paas vendors. In order to switch vendors, developers need to heavily alter the application or rebuild it. *Data Security* is also an important concern for PaaS users. As the application data or databases are stored by PaaS provider, it will be hard for users to trust PaaS vendors. A user's high *dependency on provider* might be a cause of concern as a diminutive change in vendors internal process/infrastructure might affect the functioning and performance of developers application significantly. Also, if a vendor increases the prices in their pricing model, then the application may suddenly become more expensive to use.

3.2.3 Advantages and Disadvantages of Saas

Firstly, as software is already pre-installed with SaaS services, it reduces *installation time and saves memory*[17]. Secondly, *cost reduction and easy accessibility* can be achieved by opting for SaaS as users have to subscribe the software on a monthly or yearly basis. Additionally, SaaS reduces maintenance costs as users do not need to buy external storage devices. Thirdly, *less user responsibility and workload* is provided to the user as vendor is responsible for upgrading and maintaining the software and hardware, thereby removing user's workload and responsibility. Lastly, there is a *option for data backup or recovery* for the users in case the data is lost, as the service providers install and store the data in the cloud storage in a remote location.

SaaS has two major limitations. The first limitation of SaaS is that *data is less secure* as SaaS applications are provided over internet and data security is provided by service provider[17]. The second limitation is that users *need a good internet connection* to operate the software as the software can only be accessed over the internet.

3.3 Critical Analysis

Nearly 20 years after the concept of cloud computing was proposed, it overturned the original IT industry. Cloud computing is further making the computing and

storage resources of society convenient and reasonable. Some of the new trends seen in cloud computing include computing at the edge of the network outside of centralized data centers, re-designing/re-configuring data centers for optimized execution of workloads, and using heterogeneous processors (e.g. hardware accelerators in the cloud) for faster application execution[27]. Meanwhile, 5G networks with faster speed, stable signal, and secure transmission is constructed and perfected, it will further accelerate the development of cloud computing.

In the future, due to excessive increase in the five V's of big data (variety, velocity, volume, value, and veracity) and the cloud services operators, the cloud is going to expand and change significantly. The modified cloud will be more flexible and scalable than it is today as it will posses more features that will provide more storage capability, cheaper and improved cloud services, enhanced internet services, top-notch data security, and control over data centres[28].

4 Big data in cloud computing

There are many big data tools and frameworks that are compatible with the popular cloud computing platforms of today. Several examples of these frameworks include those provided by Apache, including Hadoop HDFS/MapReduce, Spark, and Flink. Some tools such as Impala are based on NoSQL, referring to non-relational databases. Examples of these are MongoDB and HBase. With such a large variety of these frameworks available to use online, it may prove difficult to select one for a particular use case. In this section, we will first describe and then compare the performance of several popular big data processing tools, followed by a critical analysis of the results.

4.1 Popular frameworks

4.1.1 Hadoop MapReduce

Hadoop MapReduce[7] allows for the parallel execution of code by using only two functions, a *map* function and a *reduce* function. The map function creates intermediate key/value pairs based on some initial input, which are also key/value pairs. The reduce function then takes the intermediate keys and merges them in order to produce output. An example of a simple MapReduce function is one that counts the number of occurrences of unique words in a piece of text. MapReduce is known to be efficient when a single pass over the data is required, but tends to suffer when multiple passes is needed, or when the situation calls for real-time data processing[6]. A typical case where multiple iterations of a data set is common, and where MapReduce is not as effective, is when running machine learning algorithms.

4.1.2 Apache Spark, Impala, and Hive

MapReduce only operates on data stored on physical disk space, which is known to be quite slow in comparison to operating on data stores in volatile memory[24].

One of the improvements Apache Spark makes over MapReduce is the use of Resilient Distributed Datasets, which are stored in volatile memory, such as Random Access Memory. Spark uses a minibatch streaming approach and also includes many additional built-in functions on top of map and reduce, as well as API's for machine learning[6, 18].

Impala and Hive use the Hadoop environment as well as Analytic Database Management Systems instead of relational ones[6]. The result of this is a boost in speed due to their ability to handle large volumes of data in parallel.

4.1.3 Apache Flink, Storm, and Samza

Apache Flink is a framework that uses a pure streaming approach, compared to Spark's minibatch streaming[18]. It also contains a large set of built-in functionality like Spark does. Flink works by using JobManagers as a scheduler and TaskManagers to execute the jobs, with each job being handled by an individual thread.

Apache Storm is similar to Flink in that it also pure streaming, and contains master (nimbus) nodes and worker (supervisor) nodes. The nimbus acts as a load-balancer and monitoring tool while the supervisors execute one or many tasks, which may also spawn additional tasks. Storm processes data in a streaming fashion by using *sprouts* and *bolts*. Spouts process data gathered by external data streams, while bolts carry out the processing tasks.

Apache Samza is a streaming tool that uses Hadoop and Apache Kafka. Samza jobs contain one or more tasks that run in Samza containers. The jobs also consist of one or more *Kafka Topics*, which are basically data streams of a certain related type[18, 19]. For each topic, there exists a *log file*, which contain messages from particular Kafka Topics. These messages are set to live for a pre-determined amount of time.

4.1.4 MongoDB and Apache HBase

These frameworks are two examples of NoSQL databases, and are known for their security features and ability to efficiently handle large volumes of data[20]. With MongoDB, sharding and indexing features are a performance booster. Data is represented in single JSON-like structures and provides flexibility (easier to alter the schema), polymorphism (supports various data structures), and extensibility (supports various data models)[21].

Apache HBase is a popular tool for use with big data specifically, as it uses HDFS and is efficient at processing very large volumes of data[20]. Like MongoDB, HBase supports sharding of large datasets. It is also specialized for running applications with heavy random read and write operations.

4.2 Comparing performance

Several papers were reviewed, in addition to the aforementioned papers, in order to better understand how these frameworks compare to one another. One

study in particular compared most of these frameworks to each other[18]. They concluded that Hadoop MapReduce, which uses HDFS disk storage, worked considerably well with respect to using high volumes of distributed data. Spark was found to perform at a high level with iterative/machine learning tasks. Apache Flink, Storm, and Samza are built primarily for processing real-time streams of data. Similar to Spark, Flink uses volatile memory, but is also able to make use of persistent memory to avoid application crashes. Storm was initially designed with node-scalability in mind, but application performance tends to suffer during the actual scaling process. Samza was determined to have average performance overall.

Another study compared the Spark, Impala, and Hive frameworks on both large and small datasets[6]. They showed that Hive was almost always outperformed by Spark and Impala, which both performed significantly better in nearly all test cases and had comparable results to each other. Where Hive shined was when the amount of available memory was restricted (2GB). In this case, Hive was the only framework that did not crash; however, this study also illustrates that given sufficient memory (10GB in this case), Spark fairs better than both Impala and Hive. In the tests comparing performance while executing varying levels of concurrent processes, each of the three frameworks maintained consistency when scaling. Impala was shown to have the best average response times overall, with nearly no change in latency between one and 13 concurrently running processes.

Both Spark and Flink are well known competing frameworks[22]. The researchers in this study compared the performance by testing *batch workloads* (word count, grep, and tera sort) and *iterative workloads* (K-means, page rank, and connected components). It was determined that neither is the best choice for a given set of constraints. In certain cases, Spark outperformed Flink, and vice versa. One of the advantages pointed out for using Flink is that it is easier to set-up, and also uses less resources overall in comparison to Spark.

5 Discussion and Conclusion

As Sandra Harding said in 1991, what science becomes in any historical era depends on what we make of it[25]. The curiosity of what the analysis and result of big data in cloud computing will become is allowing individuals and businesses to continue exploring the field. The emergence of big data is the result of a combination of massive volumes of data and ever-increasing computing power. Specifically, mobile internet and the *Internet of Things* produce a massive amount of data. Big data computing technologies help solve the problems of massive data collection, storage, calculation, and analysis.

It can be said that big data and cloud computing work well together; however, using cloud computing to deal with big data also brings some problems and challenges. The first is security: if an organization's data is confidential or proprietary, or they need to adhere to strict security protocols, there may be concerns about storing and operating on big data in the cloud. As the cloud

is an open platform, it may be maliciously attacked by insiders and outsiders, thus the need to protect data security and privacy in the cloud has become a key issue. Second is the ever-increasing number of cloud users. Cloud service providers must address the issue of making the requested data available to users to deliver high-quality services[26]. The use of cloud computing refers to the reliance on these service providers, who may inadvertently limit flexibility and innovation, or begin to monopolize the market.

When considering which frameworks to choose for running big data tasks in the cloud, one must carefully consider which is best for their particular use case. Using Hadoop HDFS-based frameworks might not be a good idea when running iterative and/or machine learning tasks. If the amount of available memory is limited, Hive might be the better choice over Spark. When real-time data-stream processing is required, Flink, Storm, or Samza may be the optimal solution. As big data and cloud technologies evolve, it is critical to update oneself on the state of the field in order to choose the right tools and/or frameworks.

According to industry expert Michael Corrado, a World Wide Marketing Manager with Hewlett Packard, the future of cloud computing will be a hybrid IT solution (i.e. the unification of cloud-based software services and computation done on-premise)[29]. This will help private data centers in maintaining the equilibrium between the scalability and flexibility linked to the cloud, security, and control. Other industry experts, including Vice President Keff Fisher of Kemptechnologies, accord with Michael's prediction of cloud computing. We are also in assent with Michael Corrado's prediction as future data will generate with increasingly high volume, and will become arduous to store with high levels of security. Therefore, the cloud of the future will provide top-notch data security and control with scalability and flexibility. Most enterprises and organizations will need places to store their data and information securely. Cloud computing services operators are increasing daily and will only increase in the future. Therefore, there will be large competition between cloud providers for data centers, and cloud services providers will have to lower their pricing in order to remain competition. In the future, cloud service providers will need to provide higher levels of security to prevent cyber-attacks as data becomes even larger and more intricate than it is today[28].

References

[1] Big Data: What it is and why it matters. https://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[2] What Is Big Data? https://www.oracle.com/big-data/what-is-big-data.html

[3] Irfan Ahmad. (2018). How Much Data Is Generated Every Minute? https://www.socialmediatoday.com/news/how-much-data-is-generated-every-minute-infographic-1/525692/

[4] Twitter Usage Statistics. https://www.internetlivestats.com/twitter-statistics/

[5] Alexandru Adrian TOLE.(2013) Big Data Challenges. Database Systems Journal. vol. IV.31

[6] Ismail F.N., Woodford B.J., Licorish S.A. (2019) Evaluating the Boundaries of Big Data Environments for Machine Learning. In: Liu J., Bailey J. (eds) AI 2019: Advances in Artificial Intelligence. AI 2019. Lecture Notes in Computer Science, vol 11919. Springer, Cham

[7] Hashem, Ibrahim Yaqoob, Ibrar Anuar, Nor Mokhtar, Salimah Gani, Abdullah Khan, Samee. (2014). The rise of "Big Data" on cloud computing: Review and open research issues. Information Systems. 47. 98-115. 10.1016/j.is.2014.07.006.

[8] Microsoft Azure. What is cloud computing? https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

[9] Sfkp, computer encyclopedia, Cloud Computing Development. https://developpaper.com/sfkp-computer-encyclopedia-cloud-computing-development/

[10] Sushil Bhardwaj, Leena Jain, Sandeep Jain. (2010). CLOUD COMPUTING: A STUDY OF INFRASTRUCTUEW AS A SERVICE(IAAS). International Journal of Engineering and Information Technology.IJEIT 2010, 2(1),60-63

[11] Santosh Kumar, R.H.Goudar.(2012). Cloud Computing - Research Issues, Challenges, Architecture, Platforms and Applications: A Survey. International Journal of Future Computer and Communication, Vol. 1, No. 4, December 2012

[12] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang and Q. Li. (2009). Comparison of Several Cloud Computing Platforms. Second International Symposium on Information Science and Engineering, Shanghai, 2009, pp. 23-27. doi: 10.1109/ISISE.2009.94

[13] Stephen Watts, Muhammad Raza. (2019). SaaS vs PaaS vs IaaS: What's The Difference and How To Choose. https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/

[14] Kelsey Taylor. Advantages and Disadvantages of IaaS. https://www.hitechnectar.com/blogs/advantages-disadvantages-of-iaas-explained/

[15] Natallia Sakovich. IaaS vs. PaaS vs. SaaS: What's the Difference?. https://avataracloud.com/what-are-the-pros-and-cons-of-iaas/

[16] Tanmay Terkhedkar. (2019). Pros and Cons Of "Platform As A Service".https://medium.com/@tanmayct/pros-and-cons-of-platform-as-a-service-6e740ab07abf

[17] Tanmay Terkhedkar.(2019).Pros and Cons of Software As a Service. https://medium.com/@tanmayct/pros-and-cons-of-software-as-a-service-f8e05567afa5

[18] Ullah, Saeed Awan, Muhammad Khiyal, Malik. (2018). Big Data in Cloud Computing: A Resource Management Perspective. Scientific Programming. 2018. 1-17. 10.1155/2018/5418679.

[19] https://techbeacon.com/app-dev-testing/what-apache-kafka-why-it-so-popular-should-you-use-it

[20] Vishwakarma, Prasant. (2018). Comparitive Performance Analysis of MongoDB and HBase on YCSB. 10.13140/RG.2.2.15661.95201.

[21] https://www.mongodb.com/collateral/mongodb-architecture-guide

[22] Ovidiu-Cristian Marcu, Alexandru Costan, Gabriel Antoniu, María S. Pérez-Hernández. Spark versus Flink: Understanding Performance in Big Data Analytics Frameworks. Cluster 2016 - The IEEE 2016 International Conference on Cluster Computing, Sep 2016, Taipei, Taiwan. hal-01347638v2

[23] Nick Cupery. (2020). Complexity as a Service: Let Sanity Solutions help you take the complexity out of cloud solutions. https://www.sanitysolutions.com/complexity-as-a-service-let-sanity-solutions-help-you-take-the-complexity-out-of-cloud-solutions/

[24] https://www.storagereview.com/introduction-to-ram-disks

[25] Harding, Sandra. Whose Science? Whose Knowledge?: Thinking from Women's Lives. Cornell University Press, 1991.

[26] Santosh Maijhi, Gyanaranjan Shial. (2015.08). Challenges in Big Data Cloud Computing And Future Research Prospects: A Review. The Smart Computing Review. 10.6029/smartcr.2015.04.010

[27] Varghese, B, Netto, M, Llorente, IM, Buyya, R. (2020). New generation cloud computing. Softw: Pract Exper. 2020; 50: 803– 804.

[28] https://data-flair.training/blogs/future-of-cloud-computing/ , 2019.23.02

[29] https://www.futureofeverything.io/future-of-cloud-computing/

6 Contribution

Table 1: Contribution

Name	Tasks
Yuxue Liu	Introduction
	Big data
	Cloud Computing
	Discussion and Conclusion
Shane Minnema	Big data in cloud computing
	Discussion and Conclusion
	Proofreading
Mansi Mundra	Cloud Computing
	Discussion and Conclusion

Literature review of selected NoSQL and NewSQL globally-distributed database systems

Milosz Blaszkiewicz **Mostafa Doroodian**

Chuyi Tong

Web Services and Cloud Based Systems,
University of Amsterdam

Abstract

This literature review presents the globally-distributed database systems, their evolution, elementary characteristics and design trade-offs from the scientific perspective of published research. This article focuses on introducing selected systems from their scientific publications and discusses trade-offs among consistency, availability and latency that are specific to the systems operating on a global scale. Moreover, a brief discussion of related NewSQL surveys of those systems is also presented.

1 Introduction

Databases have nearly always been an inherent element of the software architecture landscape. The capability to store and retrieve data in the most efficient way absorbed some developers, architects, and researchers already since the mid-1960s when early data models were proposed to handle data storage.

1.1 Traditional databases

Not long after initial attempts, the first major idea came into existence and has since dominated the field - the relational database management systems (RDBMSs). The whole idea is based on a notion of tables and relationships binding them together. However simple it may sound, this proved to be a very durable and extensible solution, which continues to be a default choice up until nowadays. One of the main concepts within the relational databases (RDBs) is called a transaction, which is closely related to the **ACID** properties (four attributes of transactions performed in the system: atomicity, consistency, isolation, and durability). Those characteristics assure that the database is coherent and valid even in the event of errors occurring at system nodes.

Turn of the centuries brought a multitude of new challenges and, perhaps more importantly, expectations to the computing and software development. Web 2.0 brought a wave of new users and applications to the field, increasing demand for both storage and processing of large amounts of data. Those rapid changes ultimately demonstrated the shortcomings of the relational model and forced companies to look for alternatives that could offer more than the previous standard. One such idea, although actually not new at all, was a concept called NoSQL.

1.2 NoSQL

NoSQL promises straight-forward scalability, efficiency, and high flexibility. Those characteristics make this category of systems very suitable to use in clouds environments and certainly more fitting

than traditional RDBs. The term is applied rather broadly and includes various types of databases, for instance, document-oriented (like MongoDB), graph-based (neo4j), or object-based (ObjectDB). More specific definitions that were used in the early days were dropped for a more inclusive approach, which could be summarized in the new meaning of the name: *NoSQL* would only mean that SQL-type queries are not as significant as they were in RDBs.

However, what generally binds them together (to various extent) is that all those advantages come at the price of losing support for full ACID transactions. Lack of these properties rendered these systems unfit for critical or real-time applications; they also require more work on handling potential errors. Less strict systems moved more responsibility to their users. These issues, in turn, created space for yet another approach to databases.

1.3 NewSQL

NewSQL is a fresh approach to relational databases, which promises to deliver the best of both worlds: high scalability and fault tolerance of NoSQL DBs as well as ACID guarantees of the old RDBs. This means bringing back the relational and semi-relational model, however, this time, it would also provide the features to accomodatae scalability and transactional nature of operations at the same time. Such approach removed the necessity of inconsistency handling from the programmers, as well as leveraged the advantage of SQL techniques, which were heavily invested in before.

NewSQL systems sometimes do not use relational schemas to store data, but use approaches based on other models and only share relational abstraction to the users. More strict definition limits the NewSQL DBs to ones with "lock-free" concurrency control scheme and shared-nothing distributed architecture (Pavlo and Aslett [2016]).

1.4 CAP theorem and replication

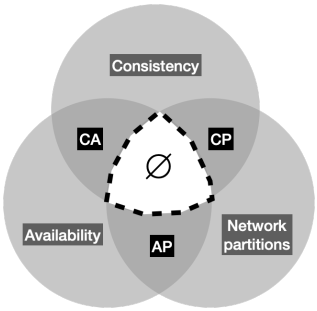


Figure 1: CAP Theorem diagram.

CAP (Consistency, Availability and Partition tolerance) theorem states that in distributed database systems only two of these three properties may be held simultaneously (Brewer. [2000]). In other words, in the presence of partition tolerance – i.e., when the system continues to operate in event of failures of the network and is a required property in distributed systems, designer has to choose between consistency – as in only the most recent state is returned, and availability – i.e., every requests receives a response.

The theorem implies that in presence of the network partition, there is no possibility to achieve both other properties in the same time. Therefore, the designer or software architect must take into account whether given DBMS prioritizes high availability or data consistency and decide based on their particular needs.

1.5 Replication

This is one of the key issues of the distributed systems. There are many reasons to introduce data replicas: an increase in reliability of the system, heightening of fault tolerance and durability of the system. Generally can be done in following variants: master-slave (designated nodes take writes and distribute them to slaves which store particular replicas), multi-master (all nodes take writes and propagate to other maser nodes which hold a replica) and masterless (very similar to multi-master,

notion of a "master node" is completely phased out) (Grolinger et al. [2013]). Other differences are in the ways systems handle synchronization between the nodes. It can be done in an eager way - when replicated data is propagated before acknowledging success to the user - as well as lazy way - first user is informed of a successful transaction, then data is replicated (lazy way is popular across NoSQL systems due to large delays from latency in waiting for acknowledgements).

It's important to note that replication has to be viewed from different perspectives. I.e., performance has to be considered, since larger number of replicas, while providing more resilience, will also add to the complexity and efficiency of the system operations; requirements in the consistency and fault tolerance layers will impact the ways the replicas are kept in the same state.

1.6 Review scope

In this review, we intent to focus on selected database systems that have been created to accommodate the needs of a deployment on a global scale, the trade-offs that have to be considered in such systems. We also want to present a comprehensive survey of existing research on NewSQL systems, which seem to be well-suited for operations on such a scale.

2 Methods

This section briefly describes how articles for this review were selected, starting from defining the subset of analysed databases. The we discuss practical assumptions for the querying search engines for our needs and present a general overview of the identified papers and of the formulated research questions.

2.1 Databases selected for comparison

A considerable number of available solutions within boundaries of NoSQL and NewSQL categories makes it necessary to select only the most impactful ones that are designed with for the globally-distributed deployments. Following the practice already proposed in other reviews, we chose to select the database management systems that are not only popular but are also visibly present in the existing scientific research.

As for the popularity metric, we sided with the well-known DB-Engines Ranking[1], which provides a monthly classification of the DBMSs based on the internet mentions, data from search engines, mentions in the technical discussion boards, job offers and even social networks.

To approximate the interest of the scientific community, we chose to perform initial searches in the ACM Digital Library and IEEE Xplore databases and use the number of publications as an indication. On average, the number of articles on the NoSQL databases is significantly larger than on the NewSQL, hence only lack of research would be considered an obstacle for databases in the second group.

2.2 Search procedure

Our article query was done in two parts due to the review constraints. First, we wanted to select at least 10 peer-reviewed articles, therefore we decided to use ACM Digital Library, IEEE Xplore and Google Scholar search engines to identify potential articles. Keywords used to find articles in this area include: *cloud-native databases*, *globally-distributed databases*, *NewSQL*, *consistency trade-offs*, *database replication*, *Google Spanner*, *Azure Cosmos DB*, *VoltDB*, *CockroachDB*, etc.

Initially, we wanted to focus our query only on relatively new papers, created at least after 2016. Unfortunately, such arrangement excluded many interesting pieces from the first half of the decade, when many defining articles were published. Therefore, we divided our query into two phases: first, we looked for the latest research papers, published after 2016 and tried to identify as many useful articles as possible, only then proceeding to the articles published before 2016.

To obtain the full picture, we also queried sources of less scientific nature, like blog posts, keynote presentations and web articles through Google search engine. Some interesting voices in the industry

[1] https://db-engines.com/en/ranking

sometimes are only present in the such forms. In this case we used similar strategy as before, focusing this time on keywords that relate to specific technologies.

2.3 Query results

Results of performed queries display a large variability in the number of existing research pieces among specific database systems. Such conclusions should not come as a surprise: database systems deployed on a global scale are a domain of large companies and strictly commercial projects. As those organizations gradually discover and accept research openness as a valuable aspect of systems development, more and more often creation of those systems is accompanied by appropriate articles published in journals or at the conferences. Very often, however, especially in smaller companies, the priorities are a little different and then available research is much less extensive. For those reasons, finding appropriate articles was not an easy task, and sometimes was totally unsuccessful.

Scientists developing and exploring these systems apparently not so often describe their findings or experiments concerning separate, closed systems. Therefore, the base knowledge sometimes had to be inferred from articles and information published in less scientific ways, such as technical blogs, general white papers and keynotes at technology-oriented conferences (e.g., CockroachDB or NuoDB).

Google Spanner, systems built on top of it and even its individual elements have relatively highest number of publications describing their inner implementations. Authors of the system published their findings by themselves, and additionally researchers added their perspective in two or three papers (like Malkhi and Martin [2013] or Agneeswaran [2017]). Other systems are not so well documented – e.g., queries about CockroachDB yielded no concrete results in research searches).

Interesting publication was an article summarizing efforts to build a database system that would further optimize operations on the global scale. One such example is Carousel (Yan et al. [2018]), however there was only one such project, stayed in its research phase and has largely disappeared since.

Due to the shortage of publications detailing specific systems, significant amount of knowledge and insights might be inferred from reviews and surveys comparing NoSQL and NewSQL systems. Section 3 discusses exiting items and highlights potential research gaps.

2.4 Research questions

Finally, we decided on the following two research questions which guide this review:

1. What systems are suitable for the globally-distributed environment and what are the differences between them?
2. What approach to consistency vs availability trade-off in the NoSQL/NewSQL databases offers the best results for modern, cloud-based computing?
3. What issues NewSQL databases have to address in terms of replication? What options are available?

Both questions are already somewhat covered in the previous works. More or less comprehensive surveys on different levels and relating to different elements are already available in a number of publications. Our study, however, is focused exclusively on the globally-distributed systems and the trade-offs among consistency, replication and latency. This review also completes existing studies by bringing an updated overview of current state-of-the-art in NewSQL solutions.

3 Related surveys

Search queries performed to identify research articles about the NoSQL and NewSQL databases of interest revealed a number of surveys and papers treating more than one system at the time. In this section we present a short study of three of them. The deciding factor whether a survey would be included below was whether a significant part was devoted to the NewSQL systems, which seem to be more interesting from the perspective of this review.

In general, they can be divided into two groups: surveys comparing the systems on their conceptual or implementational level, and the surveys comparing their performance.

Table 1: List of related surveys of NewSQL databases

No	Title	Authors	Year	Note
1	Data management in cloud environments: NoSQL and NewSQL data stores	K. Grolinger et al.	2013	Early high-level overview of cloud-native NewSQL and NoSQL
2	What's really new with NewSQL?	A. Pavlo et al.	2016	Survey of NewSQL systems
3	Performance evaluation of NewSQL databases	K. Kaur et al.	2017	Quantitative performance comparison of the four NewSQL databases

Grolinger et al. [2013] provide the most comprehensive overview of the cloud-native NoSQL and NewSQL databases. Special attention is given to the cloud environments and how their growing presence shapes the requirements for data management systems. It is also the first such survey including NewSQL systems, which were rather a novelty at the time.

Authors selected 18 database systems (including 4 classified as NewSQL: VoltDB, Spanner, Clustrix and NuoDB). Comparisons are contained mostly to the conceptual level and are organized in the following three categories:

- "Querying capabilities"
- "Partitioning, replication, consistency and concurrency control capabilities"
- "Security features"

Each of them are summarized in a neat tables with attached explanations for each attribute.

Another survey in the same category, Pavlo and Aslett [2016], sets a little different task than doing a general summary of the NewSQL techniques. Both authors are closely involved in the early development of the concept, and their idea is to examine whether various implementations lived up to the expectations.

Finally, there are also surveys of database systems that focus on their performance aspects. One of them, Kaur and Sachdeva [2017], present the results of the performance comparison of four NewSQL systems: NuoDB, VoltDB, CockroachDB, and MemSQL. Paper briefly describes the evaluated systems, proposes a methodology to assess their performance, and presents the results. The tests concern read, write, update, and execution latency separately. On average, NuoDB performs best in all of them, followed by MemSQL.

The article suffers from several issues though. The scale of the test is rather small and selective. Certain elements of the testing environment are only briefly described, making the results not easily reproducible. Moreover, the paper lacks any analysis of the results except for general remarks about the systems. This is probably the case for most of similar quantitative surveys, since those systems are not an isolated testing environments.

4 Globally-distributed cloud-native database systems

This section focuses on the differences in the implementations of selected databases, discuss their driving concepts and components, as well as provide some additional information and context of modern cloud computing era. For convenience, table 3 lists reviewed papers: articles 1 – 5 describe Google Spanner, 6 and 7 – CockroachDB, 8 – Carousel and 9 – VoltDB.

4.1 Overview of the globally-distributed database systems

Google Spanner One of the most well-known globally-distributed databases is Google Spanner. It has been described in Corbett et al. [2013]. The paper introduces the database system as the one which was designed specifically for the global scale deployment: its availability is analysed in terms of risks such as regional disasters.

Table 2: List of reviewed articles

No	Title	Authors	Year	Subject
4	Spanner: Google's globally distributed database	J. C. Corbett et al.	2013	Google Spanner database
5	Spanner's concurrency control	D. Malkhi	2013	Methods to control concurency in Spanner
6	Spanner: Becoming a SQL System	D. F. Bacon	2017	Updated article about Spanner
7	Google Spanner: Beginning of the end of the NoSQL World?	V. S. Agneeswaran	2017	ACM Sigmoid article
8	F1: A Distributed SQL Database That Scales	J. Shute et al.	2013	Database system built on top of Spanner
9	A technical overview of Azure Cosmos DB	D. Shukla	2017	Cosmos DB architectural details
10	High availability with Azure Cosmos DB	Microsoft Docs	2020	Details on high availability and consistency in Azure Cosmos DB
11	Carousel: Low-latency transaction processing for globally-distributed data	X. Yan et al.	2018	Carousel
12	A Hybrid Partitioning Strategy for NewSQL Databases: The VoltDB Case	G. A. Schreiner	2019	VoltDB Partitioning

In principle, the concept is based on splitting data across multiple sets of Paxos state machines (introduced in Lamport [1998]). Those machines are physically running inside large datacenters, located in many locations around the world. Paxos is a collection of consensus protocols among nodes. Spanner is designed to achieve high availability by replicating data inside or outside a single continent, so that consistency and availability would always be immune e.g. to natural disasters.

Two features of this database which make it interesting are: replication configuration which enables management of data locations based on the location of the user who accesses them; another feature is external consistency for read/write and read-only transactions. Below, we are going through a more detailed introduction of Spanner features. This is rather unconventional for a review, but we decided to use those few paragraphs below to explain this defining technology better by combining insights from two papers: Corbett et al. [2013] and Malkhi and Martin [2013]. The first one is describing general implementation in the initial configuration and the second one is slightly less detailed, produced by Microsoft engineers, and provides a more condensed view on the most significant elements of the system.

Initially, the first Spanner used Bigtable (Chang et al. [2006]) data storage as a base model for Spanner. After a while, first issues were recognized, mostly from OLTP developers. Those were mostly from lack of schema system, inconsistency, etc. Google, to tackle this issue, temporally build Megastore (Baker et al. [2011]) on top of Bigtable. Then because of its semi-relational data model, they have come up with a new data model that looks like Bigtable while it keeps a timestamp which makes the data more multi-version instead of a key value data store.

Finally, they came up with an idea to make Spanner a fully-featured SQL database but instead of using PostgreSQL and other existing technologies, they created their own SQL dialect based on standard ANSI SQL and named it Standard SQL. While its implementation was not entirely a novel technique, there were some unique aspects like distributed query execution (query results obtained by parallel workers), range extraction (which server and what amount of data should be searched), and query restarts (the database tries to handle the errors internally and do the process again and again to solve the problems related to distribution).

Spanner finally changed Bigtable and SSTables stack to Ressi as a permanent solution which is better for both OLTP and OLAP developers.

Figure 2 describe the spanner universe and the figure 3 shows Spanner server software stack. Each Spanner server is responsible for around 100 or 1000 data structures.

The major problem for implementing a consistent global distributed DB was local time differentiation. Spanner tackles this issue by attaching a timestamp to each transaction. This means that if a transaction T1 is committed before T2, then T1 commit timestamp is before T2 in all nodes; it reflects

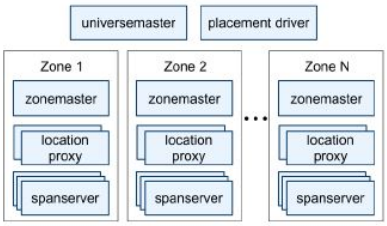


Figure 2: Spanner server organization. Source: Corbett et al. [2013]

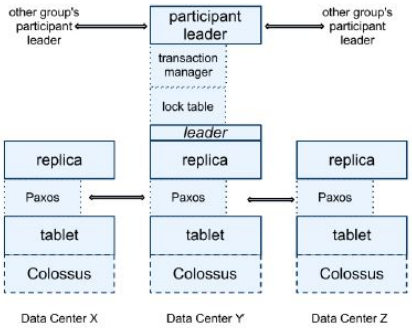


Figure 3: Spanserver software stack.Source: Corbett et al. [2013]

serialization order. To implement the timestamp property in a distributed environment, they needed to tackle clock uncertainty issue between machines, data centers, etc. So, they implemented an API named True Time which kept uncertainty at about 10ms using two hardware clock references: GPS and atomic clocks. They have used these two forms because each have its own failure modes, e.g., GPS may have antenna and receiver failures and atomic clocks can experience failures over long periods of time.

Spanner is known as a transnational database. Atomicity in read write transactions (has data lock) and read-only transactions (without data lock) are managed with a timestamp that has a lower and upper bound for transactions regarding to True Time. Spanner use wound-wind method for concurrency control. In general, committing a transaction has two-phases, preparation and committing, which means it delays the write lock until transaction is finished and if there is a deadlock – it uses wound-wind to manage it.

Due to the fact that Spanner does not have data lock for read-only transactions, reading multiple objects may cause inconsistency. Again timestamp and data versioning but keeping different version of an object will yield a consistent snapshot.

A database system that is built on top of Spanner is F1, described in detail by Shute et al. [2013]. Their key goals was to provide scalability, availability, consistency and usability for a database system serving enormous AdWords business needs. In addition to core Spanner, this DBS enabled distributed SQL queries joining data from external sources, optimistic transactions, consistent secondary indexes and asynchronous schema changes. Parts of these efforts were also included in the next version of the Spanner, released in 2017.

Bacon et al. [2017] is a paper which summarizes the evolution of the system that had taken place. Findings discussed there point to a conclusion that it has to be taken into consideration that since 2017 there were fundamental changes in the system. Their purpose was to make it resemble more a traditional database, since this seemed to be a popular demand among the users. Agneeswaran [2017] provides an external perspective on the updated system and asks a rather daring question: whether Spanner would end NoSQL databases? The answer he seems to be giving is that not exactly yet, but it has already changed a lot in the database landscape.

Microsoft Azure CosmosDB is the first globally distributed, multi-model database service. The project was started in 2010 and launched in 2017. The company released an article detailing the system Shukla [2017]. One of the most important characteristics is that it could be highly-customized due to the multi-model and multi-API solution and its added consistency choices. The system also supports low latency, high availability and high throughput, which are implemented with its global distributed system.

Microsoft Azure CosmosDB makes the tradeoff between consistency, availability, and performance. It introduces five well-defined choices of consistency instead of the legacy systems' two options, which are strong, bounded staleness, session, consistent prefix and eventual modes. Each level would leads to different availability and performance level.

As discussed in Microsoft [2020], the high availability and low latency are not only provided by appropriate consistency trade-offs level but also the data replica and partition in the same or different global regions. Figure 4 shows the replica and partition abstractions of the global distribution of Azure CosmosDB. The replica sets could be considered as physical partition here. A group of replicas comprise the replica set, realizing data consistency, durability and availability with the state machine replication approach. Partition set consists of multiple replica sets allocated globally. The membership of both of the replica set and the partition set fluctuate based on the management operations for physical partition. For the possible conflicts, including insert conflict, replace conflict and the delete conflict, Microsoft Azure CosmosDB offers two types of solution police. Solution Last Write Wins (LWW) is based on the system timestamp property while a custom defined property is also available. In the insert and replace conflict, the win item would be the one with the highest property value. In the delete conflict, the deleted version always takes the first priority no matter what property values the items have. Besides, the custom solution policy is also provided for users to configure.

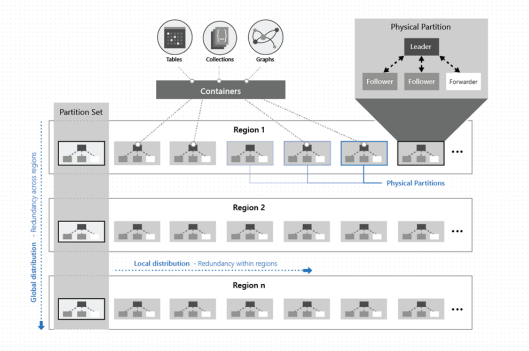


Figure 4: A distribution of resource partitions in Azure CosmosDB. Source: Microsoft [2020]

Carousel is one of the additions to the database technologies that were added slightly later than the rest described in this review. Its first version was released in 2015, and was already a work that aimed at improving certain aspects of globally-distributed databases. The founding concept of the system is achieving substantial reduction in the number of network roundtrips, which have to be performed to commit each transaction and ensure its successful completion. The architectural details and novelties introduced in this system are discussed extensively in Yan et al. [2018]. Unfortunately, the queries did not find additional research and project stayed in its research phase.

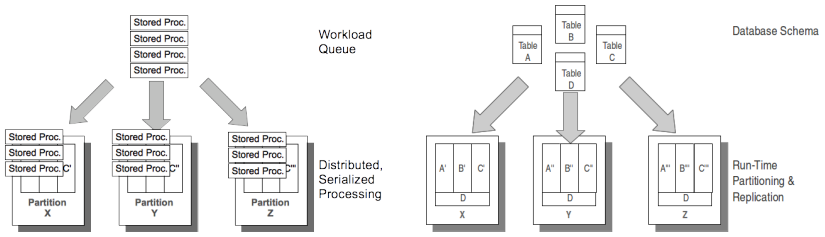
VoltDB is a commercial version of the H-Store, which was constructed at MIT in 2000s. It is a distributed in-memory relational database system, which addresses the speed with scalability, and is suitable for real-time fast data transaction analysis.

For the CAP theorem, VoltDB takes the solution of strong consistency and partition tolerant over availability, eg. when it encounters network failure it would keep done until being recovered with the same value for all the replicas. VoltDB also keeps the ACID properties to ensure real transactions.

As explained Schreiner et al. [2019], the data is stored with the in-memory approach to avoid expensive disk access, with single-threaded engine for data operation. The engine automatically operates with single-partition transactions. The serialized processing for the single-partition has ensured the consistency and reduced latency.(see Figure 5(a)) And the system runs in parallels with automatic data partitions using a logic command router for multi-partition. VoltDB also supports replication, some small tables could be joined with the stored procedures and the larger tables as well as keeping the single-partitioned transaction to improve performance (see Figure 5(b)).

4.2 Trade-offs in globally-distributed database system design

In this section we want to reflect on the approaches to consistency trade-offs choices and replication options across different database systems in a more comparative way.



(a) The overview of serialization in VoltDB (b) The overview of replication in VoltDB

Figure 5: Serialization and replication in VoltDB. Source: VoltDB Documentation

Table 3: List of reviewed articles

No	Title	Authors	Year	Subject
1	Consistency Tradeoffs in Modern Distributed Database System Design	D. J. Abadi	2012	PACELC theorem
2	Practical Tradeoffs in Google Cloud Spanner, Azure CosmosDB and YugabyteDB	S. Choudhury	2018	Blog post from a producer of another system
3	Proving PACELC	W. Golab	2018	Theoretical formulation of the PACELC theorem

As mentioned earlier, the trade-off between consistency and availability is an obstacle that has to be dealt with in all database systems. In case of globally-distributed ones, this issue is even more pressing. One such opinion is provided by Choudhury [2018] from a company creating YugabyteDB, another database system (not treated in this review due to space constraints and shortage of scientific articles). Author presents how the CAP theorem looks like in practice. Google Spanner and YugabyteDB are both classified as strongly consistent and partition-tolerant, while Azure Cosmos DB as configurable, with both options available. Paper also categorizes these three systems in terms of PACELC (discussed below): Spanner and YugabyteDB as offering low latency at the cost of lowering consistency, while Cosmos DB – again configurable. High availability in all cases is achieved through consensus protocol, which can provide also high availability.

Every database system running on a global scale has to poses the capacity to replicate its data. This, however, introduces certain additional complexity to the system operations. The way in which replication is performed, has a significant impact on the system efficiency and performance. Abadi [2012] argues that what should be actually in focus of database designers is the consistency vs. latency trade-off. He proposes a formulation of a new PACELC theorem, which integrates issues of latency added by consistency constraints into the CAP theorem. It can be summarized as: (1) when failures exist, what is the relation between availability and consistency? (2) When there are no failures, how much does the system prioritize consistency over latency? The paper lists three main methods to perform replication operations:

1. Data sent to all replicas at the same time.
2. Data sent to master nodes first, then replicated to the nodes storing replicas (synchronous or asynchronous).
3. Data sent to one location first and then is propagated to subsequent locations (also synchronous or asynchronous)..

PACELC theorem may indeed be very important in the globally-distributed systems, as the latency vs. consistency trade-off is unavoidable in systems operating over WAN networks. There is no denying to the fact that latency has an important role in the performance of systems deployed in distant datacenters; therefore, PACELC is an extension of the classic CAP theorem which has large impact on those global systems.

Formulation of the PACELC theorem initially missed an important feature of the CAP theorem: the practical concept was called a theorem mostly due to parallels drawn to the CAP theorem. More formal formulation of the theorem was done only in Golab [2018], which places the principle within

Table 4: Comparison of the database systems

No	Database	Type	Consistency	Replication
1	Google Spanner	NewSQL	Strong consistency	Replicas ordered using global timestamps, Paxos
2	Azure Cosmos DB	NoSQL	Configurable, no strong consistency like in Spanner	Replication inside regions, also configurable by users
3	Cockroach DB	NewSQL	Strong consistency	Raft consensus protocol, at least 3 nodes for high-availability
4	Carousel	NewSQL	Strong consistency	Extended Raft consensus protocol
5	VoltDB	NewSQL	Strong consistency	Updates executed on every replica simultaneously
6	NuoDB	NewSQL	Eventual consistency	Logical ordering, MVCC

broad distributed algorithms field and connects it to more theoretical works in distributed objects communication.

5 Conclusions

Globally-distributed systems emerged already a some time ago. More and more tailored solutions are being developed and used throughout the database spectrum, and this literature review presents some of them and also discusses scientific contributions to them. In this review, we provided an overview of globally-distributed database systems. We also identified the practical trade-offs present in such distributed databases, the CAP and PACELC theorems, as well as discussed them in terms of selected databases.

Unfortunately, traditional research into this field is largely limited. There are two main reasons for why this is the case. First and foremost, databases operating on a global scale are a domain closed to large companies, where such solutions are commercial projects; therefore, sharing the insights and findings is much more complicated matter than within academia. Secondly, testing and experimenting on systems that are running as a part of a large globally-deployed systems, is guaranteed to be error-prone. Many aspects of those systems are closed and unclear to the outside parties – i.e., investigating particular aspects of those systems would very often require at least partial cooperation with the their operators, leading again to the problems mentioned above.

More independent research efforts, for instance Yan et al. [2018], have to, by their nature, be more small-scale and therefore have to focus on specific issues present in this field. Mentioned paper shows that such undertakings might be completed successfully and lead to interesting findings.

Another category of articles mentioned in this review are surveys and reviews concerning NoSQL and NewSQL systems. NoSQL systems, existing for a longer time, are surveyed much more often than NewSQL ones. Again, this disproportion may be credited to the fact that NewSQL systems are more often deployed on significantly larger scale and require much more resources, while NoSQL can work in such environments, but not necessarily have to. As a matter of fact, there are no surveys focusing on the aspects that are most important in the globally-distributed setting. Our queries found only one quantitative survey of NewSQL databases, comparing limited number of systems in terms of latency. There is certainly space for more comparative experiments between those systems. The question, however, remains, how much such experiments can reveal without using appropriate resources to run them in a controlled environment on a global scale.

Importance of applications operating on a global scale has been rising drastically already for a couple of years and seems to continue in the foreseeable future. Database systems play crucial role for those and thus will have to be developed further. Especially NewSQL systems, combining the best of relational and NoSQL databases, will most likely continue to attract attention of the technological community. We cannot underestimate also the progress made in the NoSQL systems, which perform really well where transactional nature of operations is not so important. What seems to be of incredible importance is how they cooperate in clouds environment.

References

D. Abadi. Consistency tradeoffs in modern distributed database system design: Cap is only part of the story. *Computer*, 45(2):37–42, Feb. 2012. ISSN 0018-9162. doi: 10.1109/MC.2012.33. URL https://doi.org/10.1109/MC.2012.33.

V. S. Agneeswaran. Google spanner: Beginning of the end of the nosql world?, 2017. URL https://wp.sigmod.org/?p=2153.

D. F. Bacon, N. Bales, N. Bruno, B. F. Cooper, A. Dickinson, A. Fikes, C. Fraser, A. Gubarev, M. Joshi, E. Kogan, A. Lloyd, S. Melnik, R. Rao, D. Shue, C. Taylor, M. van der Holst, and D. Woodford. Spanner: Becoming a sql system. In *Proc. SIGMOD 2017*, pages 331–343, 2017.

J. Baker, C. Bond, J. C. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M. Leon, Y. Li, A. Lloyd, and V. Yushprakh. Megastore: Providing scalable, highly available storage for interactive services. In *Proceedings of the Conference on Innovative Data system Research (CIDR)*, pages 223–234, 2011. URL http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper32.pdf.

E. A. Brewer. Towards robust distributed systems, July 2000. URL https://people.eecs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf.

F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 205–218, 2006.

S. Choudhury. Practical tradeoffs in google cloud spanner, azure cosmos db and yugabytedb, 2018. URL https://blog.yugabyte.com/practical-tradeoffs-in-google-cloud-spanner-azure-cosmos-db-and-yugabyte-db/.

J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.*, 31(3), Aug. 2013. ISSN 0734-2071. doi: 10.1145/2491245. URL https://doi.org/10.1145/2491245.

W. Golab. Proving pacelc. *SIGACT News*, 49(1):73–81, Mar. 2018. ISSN 0163-5700. doi: 10.1145/3197406.3197420. URL https://doi.org/10.1145/3197406.3197420.

K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz. Data management in cloud environments: Nosql and newsql data stores. *J. Cloud Comput.*, 2(1), Dec. 2013. ISSN 2192-113X. doi: 10.1186/2192-113X-2-22. URL https://doi.org/10.1186/2192-113X-2-22.

K. Kaur and M. Sachdeva. Performance evaluation of newsql databases. In *2017 International Conference on Inventive Systems and Control (ICISC)*, pages 1–5, 2017.

L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998. ISSN 0734-2071. doi: 10.1145/279227.279229. URL https://doi.org/10.1145/279227.279229.

D. Malkhi and J.-P. Martin. Spanner's concurrency control. Technical Report MSR-TR-2013-36, September 2013. URL https://www.microsoft.com/en-us/research/publication/spanners-concurrency-control/. ACM SIGACT News, Distributed Computing Column 51.

Microsoft. High availability with azure cosmos db, 2020. URL https://docs.microsoft.com/en-us/azure/cosmos-db/high-availability.

A. Pavlo and M. Aslett. What's really new with newsql? *SIGMOD Rec.*, 45(2):45–55, Sept. 2016. ISSN 0163-5808. doi: 10.1145/3003665.3003674. URL https://doi.org/10.1145/3003665.3003674.

G. A. Schreiner, D. Duarte, G. Dal Bianco, and R. d. S. Mello. A hybrid partitioning strategy for newsql databases: The voltdb case. In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications Services*, iiWAS2019, page 353–360, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450371797. doi: 10.1145/3366030.3366062. URL https://doi.org/10.1145/3366030.3366062.

D. Shukla. A technical overview of azure cosmos db, 2017. URL https://azure.microsoft.com/en-us/blog/a-technical-overview-of-azure-cosmos-db/.

J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina, S. Ellner, J. Cieslewicz, I. Rae, T. Stancescu, and H. Apte. F1: A distributed sql database that scales. In *VLDB*, 2013.

VoltDB Documentation. Voltdb documentation: How voltdb works. URL https://docs.voltdb.com/UsingVoltDB/IntroHowVoltDBWorks.php.

X. Yan, L. Yang, H. Zhang, X. C. Lin, B. Wong, K. Salem, and T. Brecht. Carousel: Low-latency transaction processing for globally-distributed data. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 231–243, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450347037. doi: 10.1145/3183713.3196912. URL https://doi.org/10.1145/3183713.3196912.

Group members contributions to the Review

Table 5: Tasks and people completing them

No	Task	Person responsible
1	Sharing a broad list of articles in the distributed databases domain	Mostafa Doroodian
2	Creating a refined list of research articles on globally-distributed databases	Milosz Blaszkiewicz
3	Writing Introduction, Methods, Related Surveys sections	Milosz Blaszkiewicz
4	Writing Google Spanner part, contributing to Replication definition	Mostafa Doroodian
5	Writing Azure Cosmos DB and VoltDB parts	Chuyi Tong
7	Writing Carousel, Trade-offs in globally-distributed database system design and Conclusions parts	Milosz Blaszkiewicz
8	General corrections and coordination	Milosz Blaszkiewicz

IDaaS (Identity as a Service): Challenges and visions for the future

Sagnik Aditya [13252836], Milica Đorđević [2667572] and Stijn Veken [12573396]
Department of Computer Science
Universiteit van Amsterdam
Amsterdam
Group 7, Web Services and Cloud Systems. May 2020.

Abstract

In the recent years, with the advent of cloud computing, there has been a sizeable shift in computing tendencies. Small and Medium Sized Enterprises are increasingly ditching traditional computing architectures and migrating to the cloud in order to cut costs, scale easily and maintain their focus on the core expertise. This shift has resulted in number of services moving to cloud, one such being identity management. This new concept is known as Identity as a Service (IDaaS). In this review, we aim to explore challenges in IDaaS and solutions proposed to address them. There is no solution that addresses all identified challenges and it does not seem to be even possible. Many solutions are feasible only in certain settings, while others require trade-offs among different challenges.

Keywords: Identity as a Service(IDaaS), federated identity management, privacy preserving, identity management, OpenID, Proxy Re-Encryption

1 Introduction

The digital society we live in today is heavily dependent on identities. They are the key enablers for communication between end-users and service providers. Moreover, identities are becoming invaluable asset for those in charge of its management. Users identities can be used for the sake of planning future business ventures, marketing purposes or they can be even sold to third parties. Therefore, it is no surprise that identity management systems (IMSs) are one of the most commonly deployed services within organisations' environments. IMSs primarily serve to protect access to information and resources that an organisation possesses. Unfortunately, when deployed on-premise, managing these systems becomes costly and time consuming. It requires specialised applications and personnel. Cloud computing gave birth to the concept of Identity as a Service (IDaaS) - outsourcing identity management in the cloud. However, there are a number of challenges that may prevent the adoption of this relatively new concept, especially regarding privacy of user identities due to the loss of control over this highly confidential data. [1]

In this review, we aim to explore the main challenges that the field of IDaaS is facing and solutions proposed so far to overcome them. The solutions are critically inspected on their feasibility with respect to all entities involved and the settings in which they may be applicable. Identifying challenges is important for readers that are considering to adopt this relatively immature concept. They shall be aware of the issues that the adoption imposes. On the other hand, both challenges and possible solutions may serve as a guidelines for readers who are thinking of providing one such service, considering both technological and economic feasibility of the proposed solutions. Finally, the brief history and overview of IDaaS aim to familiarise the general public with this concept. We are primarily focused on the solutions that address privacy issues. The rest of the review is organized as follows. We briefly introduce the main functions of a typical IMS, then we overview the important

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

identity models, both traditional ones and the ones in cloud, including the IDaaS model. We explore the early motivation and the first mentioning of IDaaS concept. We overview both benefits of adopting IDaaS and its challenges and shortcomings. In the main part of this review, we explore the proposed solutions for overcoming some of identified challenges, focusing mostly on privacy. Finally, we discuss some potential directions that the field of IDaaS may head to in the future and we draw the main conclusions.

2 Identity Management Systems

The main purpose of IMS is to facilitate creation, storage, and usage of identity information. Identities are used primarily for authentication and access control in order to protect information and resources that an organisation possesses. IMSs were traditionally designed for internal usage, within the boundaries of an organisation. However, due to the increased usage of Internet, the number of interactions that organisations have with external users is dramatically increasing. The scope of managed identities ranges from employees of an organisation to the entire world population. [1]

A typical IMS consists of processes and technologies that aim to secure an organisation's resources and information, but protect users' identities and profiles at the same time. The number of such elements may vary. According to the work of Mpofu and Staden [2], the basic elements are:

- *Directory*: defining and storing users' identity information.
- *Life-cycle management tools*: adding, modifying, and deleting identity data from the directory.
- *Regulatory mechanisms*: for regulating user access to data, usually via policies and access privileges.
- *Auditing and reporting tools*: keeping track of whom and when accessed the system.

However, there is no clear consensus over what IMS shall entail. In the work of Nuñez and Agudo [1], underlying management protocol is also qualified as essential part of IMS, even though they may be a variety of different supported protocols. They enlisted authentication and authorisation mechanisms as important parts of IMS's functionalities, yet in the same work they proposed the hybrid solution in which authentication is performed outside of realm of IMS.

3 Identity Models

The entities involved in the IMS are: a *service provider* (SP) that provides online services to *users*, while an *identity provider* (IP) manages users identities and authenticates them prior to accessing SP's services. According to the work of Zwattendorfer et al. [3], traditional identity models are classified based on the storage location of identity data:

- *Central approach*: the identity data is managed and stored in the central repositories in IP's domain. Prior to using SP's services, a user has to register at an affiliated IP. User have to be successfully authenticated at the IP before accessing any application or service offered by the SP, but they have no control over the stored identity data nor information that is being transmitted between IP and SP.
- *User-centric approach*: identity data is stored and managed within the user's domain (usually on a smart card) and they always remain the owner of it. SP can request and receive the data only with the user's prior consent.
- *Federated approach*: the user's identity data is distributed over several IPs that have established trust among each other. The trust relationships are usually established at organisational level and the repositories of IPs are linked for the sake of secure exchange of identity information.

With many applications moving to the cloud, new questions regarding identity management have emerged, primarily regarding the deployment of IP. In IMSs in the cloud, there is an additional entity involved - *cloud service provider* (CSP), which hosts the cloud application. Several cloud identity management models are presented in the same work, all of which have their benefits and drawbacks:

- *Identity in the cloud*: the CSP acts as an IP. This is the special case of central approach, with SP and IP being the same entity in the cloud environment. While in this model organisations

are unburdened by identity management responsibility, which is transferred to CSP, they completely lose control over identity data of their users.

- *Identity to the cloud*: the application is deployed in cloud, but the IP remains on-premise. This allows organisation to remain in full control over the users identities and reuse their existing IMSs at the same time. However, interoperability may be an issue if CSP does not support protocols and technologies that are needed for communication with IP. Interoperability issues are not only of technological, but semantic nature as well. The user attributes exchanged between IP and CSP may not be understood.
- *Identity from the cloud*: IP fully resides in cloud, but it is not necessarily hosted by the same CSP which hosts the application. The IP and the application are hosted and deployed separately. This is, in fact, IDaaS model, but in IDaaS the application is not necessarily hosted in the cloud. It can be deployed on-premise as well. The benefits of this model are cost reduction and less maintenance efforts for an organisation, but also the freedom of choice of desired IP. However, the organisation loses control over identity data and has to trust the third party to manage their user identities as agreed.

4 Early Motivation for IDaaS

The first appearance of the term *Identity as a Service* in scientific papers can be found in the work of Emig et al. [4], back when migration to service oriented architecture (SOA) was a popular research topic. Identity management becomes an issue when services need to be composed. Built-in IMSs in underlying services complicate the integrated view on identity management and impose interoperability issues. In the SOA spirit of loosely-coupled and independent services, the authors suggest that identity management should be also isolated in a separate service and consumed as such. In their proposed architecture, identity management service encapsulates two functionalities that are offered via interfaces - authentication and authorisation. This allows its complete separation from the core concern part of SOA. Although the separation of identity management allows the reuse of this common service in different business processes, there are no details of how such service should be reused and integrated. Administration of identity data is completely neglected with the excuse of being usually performed by humans and not automated. Nowadays, the term *Identity as a Service* is usually referred to IMS outsourced in cloud, but the initial vision of separation of IMS still remains.

Shortly after that publication, cloud computing emerged as new and promising technology. However, the highly dynamic environment, where services are provisioned and de-provisioned in real time, imposes even greater identity management problems. The issue of identity management in the cloud is explored in the work of Gopalakrishnan [5] and is out of the scope of this review. However, the author concludes that the shortcomings in identity management solutions in the cloud forced cloud vendors to consider outsourcing identity management, which gave birth to the concept of IDaaS.

5 Benefits of IDaaS

On-premise delivery of IMS, within the internal infrastructure of an organisation, introduces an overhead in cost and time [6]. Specific applications and highly skilled personnel are needed for management, integration and maintenance of such systems [6]. IDaaS offers opportunity to outsource IMS and deploy it in the cloud. Software management efforts are thus minimized since this responsibility is referred to IDaaS provider. This is *on-demand* model where identity management services are rapidly deployed and payed per use, without additional cost overhead. This utility pricing model also allows organisations to dynamically and easily scale up and down according to their needs and number of users whose identities shall be managed [2]. This is not the case with *on-premise* model in which organisations have to handle scaling by themselves and may not be able to support large number of users with their current infrastructure. Additionally, IDaaS providers may have their services deployed in disperse geographic locations, around the entire world [2]. This will reduce the communication overhead with geographically distributed users of an organisation's services, which they would unlikely solve on their own. Moreover, IDaaS providers are experts in the realm of identity management and they provide access to advanced and dedicated security tools [2]. Finally, outsourcing the complicated task of identity management to the external party allows organisations to focus solely on their core business services [2].

6 Challenges in IDaaS

The problems with IDaaS are inherent to cloud computing problems - loss of control over the outsourced data. However, in case of IDaaS, the data represents user identities which are protected by specific regulations (for EU by European Data Protection Directive). This raises the issue to another level. Cloud providers define service level agreements (SLAs) and internal security policies to address such problems, but nothing stops them from breaking the agreement. Users simply trust them not to do so. The main challenge in IDaaS is allowing users to enjoy all the benefits of cloud computing, but keeping them in control of their data and preserve their privacy at the same time. [6]

Privacy is just one of the challenges in IDaaS. There are number of other issues, all triggered by the issue of trust. These issues are surveyed in the work of Mpofu and Staden [2]. They define *trust* as confidence that something will be surely delivered as promised. The issues are further classified as *institutional*, *character* and *loss of control* issues, based on the source of trust.

Institutional trust issues are related to societal structure of an institution that provides IDaaS (referred as IP in the rest of this section). These include:

- *Competence*: the capability of a provider to implement functionalities of IMS. This includes skills and knowledge for handling and responding to requests for identity management. If users doubt in competence of IDaaS provider, they will not adopt it.
- *Interoperability*: there are variety of platforms, protocols and approaches in IMS. Ones provided by an IP may not be compatible with users needs. The IPs shall support heterogeneity in different aspects of IMS to satisfy diverse needs of their users.
- *Dependability*: a dependable IDaaS shall be ready to provide service as specified continuously, but also undergo modifications according to the changes in the environment. Dependability spans to different attributes such as availability, reliability and maintainability.
- *Confidentially*: identity information shall not be accessed nor modified by unauthorised individuals.
- *Multi-tenancy*: when a single cloud instance of IDaaS is used by different customers, there should be no leak of identity data among them.
- *Auditability and accountability*: keeping track of access history in order to know what is happening with system and who is responsible for it.

Character trust issues are related to the behaviour of IP. These issues are refined to:

- *Fairness*: confidence of customers that they will not be taken advantage of by IP.
- *Credibility*: a way in which customer perceives an IP. It is usually based on existing customers, encounters and familiarity in general, but also perception of IP by other trusted parties.
- *Predictability*: consistency in past actions that allow customers to predict how will IP react and response on the future actions.

Loss of control issues are direct consequences of outsourcing the IMS. These issues include:

- *Ownership and control over infrastructure*: customers are not sure how secure the infrastructure where IDaaS is deployed is.
- *Control over identity life-cycle*: when, for instance, deletion of certain identity data is requested by customer, they are not sure whether the data is indeed deleted by IP.
- *Vendor lock-in*: great efforts shall be made to overcome highly probable incompatibility with certain IP, which makes it more difficult to switch to another one for whatever reasons. Standardisation is the only way in overcoming this issue and simultaneously improve interoperability.
- *Notifications and redress*: there should be mechanisms for notifying customers in case of security breaches.
- *Access and transparency*: open up and inform customers regarding every aspect of management of their identity data they wish to know.

The authors made a lot of effort to classify the issues, but many of them are interconnected or overlap. For instance, there is a clear overlap between interoperability and vendor lock-in issues. The less diverse the technologies that IP incorporates are, the bigger the chance is of adopters running into a vendor lock-in issues and vice versa. Moreover, all the issues classified as institutional can be traced

back to lack of competence e.g. providers may not be able to handle interoperability and diverse technologies because they are not competent enough to do so.

7 Solutions

In this section, we aim to explore and critically analyse the solutions proposed so far for overcoming the challenges regarding user privacy, dependability and interoperability respectfully. We identify the setting in which the solutions may be applicable, their shortcomings from the tecnical aspect, but also economic feasibility with respect to IP.

7.1 Solutions for Privacy

Preserving users' privacy is the main challenge of IDaaS. However, the term *privacy* is vague and it is usually referred to as a subset of various concepts. One such concept is *data-confidentiality* - the property of not disclosing the data to entities that are not authorised to know the data [1]. The solutions described in this section tackle the problem of data-confidentiality with respect to IP. The issue that is closely related to data-confidentiality is the fairness of IP. It is assumed that IP is *honest-but-curious* - they fulfil the agreed protocol correctly, but they may store and collect information about users without their consent. If the data can be read by IP, then one shall assume it will be read [1]. A natural solution for this problem is encrypting the data before sending it to the IP. Many solutions use *proxy re-encryption scheme*. This is an asymmetric scheme in which one party sends a message to the other via third entity, a proxy, but the proxy cannot decipher the message [1]. The proxy only re-encrypts the message using the key that was generated by the sender, while the receiver can decrypt the message using only their secret key. In the setting of presented solutions, the identity data is exchanged between a user and a SP, while IP acts as a proxy.

In the work of Nuñez et al. [6], the proposed IDaaS model integrates the proxy re-encryption scheme into the OpenID Attribute Exchange protocol. This model enables IP to serve requested user attributes to SP without being able to read their values thus preserving user's privacy. Open ID is a decentralised authentication protocol that allows SP to delegate user authentication process to IPs. In the main flow of OpenID, a user wishes to use SP's service, but needs to be authenticated prior to that. For this purpose, the user may choose one of the affiliated IP's, at which they wish to authenticate at, and the result of the authentication is transferred to the SP. In case of successful outcome, the user may use the SP's services. However, the SP may request additional user attributes (first name, last name, age etc.). The IP transmits the requested attributes along with the authentication result and this interaction is defined by OpenID's extension called Attribute Exchange. The fact that the IP manages user attributes while being able to read them is a weak point that violets the user's privacy. This is why the authors proposed integration of proxy re-encryption scheme. User attributes are encrypted with their public key and stored at the IP. When SP requests the attributes, the user needs to transmit re-encryption key to the IP so that IP can perform the re-encryption on requested attributes. IP delegates the attributes to the SP which can decrypt them using only their secret key. This solution not only prevents IPs from reading user attributes, but it also prevents them from disclosing this data to any party other than the SP that user choose to authenticate. The needed re-encryption key is generated by the users themselves for that specific SP. However, there are a number of design issues in this model. First, the users shall trust IP to remove the re-encryption key after the authentication, which they may not do. Second, the proxy re-encryption scheme requires certain public parameters to be known to all parties prior to authentication. There should be a certified authority that IP, SP and users trust to transmit the parameters. Third, the users normally interact with SP using web-browser which are limited in their cryptographic support. This is a serious issue since attribute encryption and proxy re-encryption key generation are complex tasks that shall be performed at the user's side. The transfer of user attributes to the IP prior to described flow is not tackled in this solution. However, the IP should be at least entitled to see the names of the encrypted attributes so that those requested by SP would be transmitted. The authors completely ignored the fact that user attributes are encrypted using their public key which in theory can be done by anyone. IP needs to make sure that received attributes indeed belong to the user in question and prevent some malicious data being transmitted to SPs. From the economic aspect, the IP should only incorporate the appropriate cryptographic mechanisms to perform re-encryption. The authors analysed typical number of requested authentications in a setting of average IT portfolio and their respective users. The additional yearly costs would yield around

1749.29 USD by their estimations, which is more then a reasonable price for a small or medium sized enterprise. Still, the changes would affect both users and SPs that use their services.

The previous solution was applied to general scenario, where user and SP can be anyone. The design issues aside, privacy-preserving asset of re-encryption scheme motivated Slamanig et al. [7] to incorporate it into more specific area - eGovernment. In this field, privacy of highly confidential user identities, issued by a government's Registration Authority (RA), is of crucial importance. Their proposed model is user-centric, the identity data that RA issues to citizens is stored on a smart card and citizens always remain the owners of their data. The citizen can use this identity data to access certain SPs with prior authentication at IP. In this setting, the transmission of public parameters for proxy re-encryption scheme is not an issue. Citizens, SPs and IPs are all known to RA which can fulfil this role of trusted authority for transmission of public parameters. The usage of proxy re-encryption prevents IPs from reading citizen attributes, so they can be deployed in environments that are not regarded as secure from government's perspective. Public clouds certainly fall into this category, so other benefits of IDaaS can be enjoyed with this model. However, the issue in eGovernment field is that identity data is transmitted with all user attributes included to SP when citizens want to access their services. The identity data is signed by RA as a whole and needs to be verified as such at SP's side. The authors propose the usage of additional cryptographic schemes that will allow citizens to extract only those attributes that they want to disclose with SP without changing the RA's signature. The fact that user owns a special device for reading the smart card on their side (client-side model) or they interact with trusted government's authority to use their digital identities (server-side model) eliminates the lack of cryptography support on user's side that existed in the previous solution. However, the issue of trusting IP to remove the re-encryption key remains. The authors argue that their solutions does not require a prior registration at RA of neither IP nor SP for the usage of government's identities for user authentication purposes. However, the lack of registration would remove the benefit of public parameter transmission between parties.

In the work of Nuñez and Agudo [1], proxy re-encryption is used for resolving data confidentiality issue with respect to IP in another more specific setting - federated identity management. Federated models aim to resolve the issue of identity fragmentation - the user identities being distributed over different domains. The users are identified at each SP separately. This decreases usability with respect to users since they have to manage various credentials imposing security risks due to password reuse. In a federated model, user identities are transferred across different domains among organisations that have established relationships of trust. The employees of an organisation want to access a SP with whom organisation has established trust, but they need to be authenticated at IP first. The authors propose BlindIdM model which uses SAML as underlying protocol due to its mechanisms for establishing trust and extensibility. The latter is important because user attributes are encrypted using strong, symmetric algorithm (e.g. AES) and stored at IP, while the symmetric key is encrypted using proxy re-encryption scheme. This complex information should be reflected in response, so SAML's extensibility is of crucial importance. While size of a user attribute may vary, the length of a symmetric key is fixed, so the efficiency of proxy re-encryption scheme is improved. The authors opted for a hybrid IDaaS approach - attributes of employees are stored and managed at IP, but authentication is held on-premise, by the organisation to which the employee belongs to. They did not state the reasons for choosing the hybrid approach, but presumably for improved user privacy with respect to IP. However, the authors neglected that this approach increases a number of interactions needed for authentication. An employee is redirected from SP to IP, then from IP to their host organisation which performs authentication, then again to IP for collecting requested attributes and finally to SP. This certainly has a negative impact on performance despite the improvement regarding fixed-length key re-encryption. The authors explored additional costs of incorporating proxy re-encryption scheme at IP and the estimation is the same as in previously presented OpenID model. The IP needs only additional cryptographic support for re-encryption purposes. The interesting question that arose is why would an IP even want to implement a BlindIdM model. They would not only have additional costs, but lose an important asset - user identities, which they cannot access anymore. IPs can use the data confidentiality attribute as an added value of their product and gain more appeal to the customers. Additionally, they will be less liable in case of misuse of user attributes since they are not able to access them in the first place.

The choice of hybrid approach in the previous solution raises a question if authentication and authorisation should be handled on the application's side or at the IP. The SPs certainly have more control if they perform these functionalities themselves, but their operational costs would significantly

increase. The SP has to support different number of their expected users which they may not be able to do with their own infrastructure. This means that SPs lose not only cost benefits of IDaaS, but also ease of scaling. Moreover, if the IP is in charge of only storage and management of identities, but not authentication and authorisation, it is not clear if this hybrid approach really improves privacy to any extent. The IP still has the user identities on their side and the privacy can be diminished regardless of whether they perform authentication and authorization or not. In the work of Vo et al. [16], it is proposed to remove the identity propagation out of the application implementation by putting IDaaS in control of the Authentication and Authorisation Infrastructure (AAI) life cycle (i.e. Provisioning, Establishment of trust and Termination in a target platform). In contrast to describing security policy in a platform specific manner, this option makes it platform independent. The security layer would be added as a separate virtualization layer. This increases adaptability of applications and reduces the technical knowledge required to implement a security layer in applications. The main benefits of this solution would be reduced operational expenditure, high scalability and flexibility, and reduced complexity of the technological stack. However, the issues of up-time, accessibility and response time are not tackled in this paper. We argue that this can be overcame with the deployment across multiple data centers in disperse geographical areas. In case that one of the data centers is attacked, the service could retrieve the relevant data from other active data centers thereby ensuring up-time and accessibility. Solution for the increased dependability will be discussed in the upcoming subsection.

7.2 Solutions for Dependability

Dependability challenges are directed towards achieving high availability and resilience to malicious attacks. IPs store and manage extremely confidential user identity data which makes them single points of failures and main targets of attacks. If an organisation's IMS fails, then all their resources and information become compromised and non-reachable. On the other hand, advanced threats, such as large scale distributed DDoS attacks and data leakage, are becoming more frequent and more dangerous. In the work of Kreutz and Feitosa [8], the so called cloud-of-clouds model for deploying highly dependable and secure IDaaS was proposed to address these challanges. This is one of the rare works written from the perspective of an IP - it does not treat the IP as a black-box that interacts with other entities, but discusses its inner design. The authors argue that dependability and security should be conceptually integrated in the design of an IP, not patched with features in ad-hoc mode when it may be too late. In the proposed cloud-of-clouds model, the IP's service components are replicated and reachable by consumers via gateways. The redundancy enables high availability and resilience to attacks - in case that one replica fails, others are still reachable. However, the authors emphasise that the extent of dependability and security is highly dependent on how the components are deployed.

The most serious problem arises when all replicas are deployed in a single physical infrastructure. A malicious attack at the data-center would affect all replicas, but other incidents may greatly compromise the availability of IDaaS (e.g. the energy blackout). The alternative is to deploy replicas among different public cloud providers. There is a very low chance that such incidents would affect all cloud providers at the same time. However, this solution degrades the performance of IDaaS since it introduces communication overhead among distant data centres. Deploying replicas among different data centres of a single cloud provider would reduce performance degradation if the cloud provider has dedicated and efficient network links that connect its data centres. However, the problem that IP faces when deploying all replicas in a public cloud is privacy. Malicious sysadmin in the cloud provider could compromise the confidentiality of their customers' identities. It is interesting to notice that IPs face the same privacy issues when deploying their service in public clouds as those that their customers face with respect to them. If the IP deploys all replicas in their private cloud, the privacy issue will be resolved, but their capital and operating expenses will increase. Moreover, they would lose the ease of scaling that public cloud enables. Public cloud vendors have a large pool of resources that they can use to fight the external security attacks in much efficient way than private clouds can. A hybrid approach can serve as a compromise: the replicas can be further decomposed so that the safety-critical components of IDaaS are deployed in the private cloud and the rest in public clouds. An IP gateway is certainly a safety-critical component that should not be deployed in a public cloud. Finally, a new type of cloud can also be exploited to address performance issues - telco cloud. Telecom companies have recently realised their potential of becoming large players in cloud computing. They own flexible, scalable and dynamic network systems in their data centers that only rare public cloud providers can compete with. They used this advantage to offer a concept of

network-as-a-service (NaaS) with QoS at the network level that can efficiently and securely connect geographically disperse data centres which would help the IPs in overcoming the performance issues in the proposed model. From the economic aspect, the authors conclude that full deployment of IDaaS in private cloud is not a feasible option for small and medium size companies. This is partly due to increased capital and operating expenses when IPs have to manage the infrastructure by themselves, but mostly due to high salaries of security experts that are needed to provide re silence. In this aspect, it is hard for such companies to compete with the likes of Amazon and other public cloud vendors. Deploying IDaaS replicas in public clouds is the only economically feasible solution for small and medium size companies and this can be a win-win situation for both them and public cloud vendors. The authors experimented with cloud-of-clouds model deployment in a single data center and multiple data centres of a single public cloud vendor. Both better performance and larger number of supported users resulted in the former option.

7.3 Solutions for Interoperability

The IDaaS model proposed to enhance interoperability is so called identity broker model. The identity broker is a central entity that acts as a hub during the communication of a SP with various IPs. In this model, the SP is decoupled from IPs and it communicates exclusively with the identity broker in order to authenticate its users. The benefit of this model is the aggregation of multiple trust relationships towards different IPs into only one. Moreover, the broker hinders the technological and protocol diversities of different IPs and offers a single interface for SPs. Unfortunately, the centralised nature of this model has one major drawback - the SP is now completely dependent on the broker. If the broker fails, then all connections to different IPs fail too and the SP does not have a functioning IMS. Another disadvantage is that both SP and its users are dependant on the same identity broker. If a user wants to authenticate at IP that the broker does not support, the user cannot access the SP's services at all. These disadvantages of identity broker model are explored in the work of Zwattendorfer et al. [3] and the authors propose an enhancement to overcome these issues. They propose a federated identity broker model in which different identity brokers would cooperate in order to authenticate SP's users. The users can choose a broker they wish to authenticate at and the result of the authentication will be referred to the broker that SP interacts with. In this model, the SP and its users do not necessarily depend on the same broker which increases both flexibility and interoperability. However, a lot of requirements that this model must fulfil are left untackled. First, from the technological aspect, there should be a protocol for communication between brokers. This protocol must incorporate the previously established trust relationship between them. Moreover, the common semantics of the user attributes that are exchanged between brokers should be established so that interoperability between them is possible. This can be solved with standardization. Second, the relationship of trust should be tackled from the legal perspective: who is entitled for what and who is liable in case of incidents. Lastly, from the business perspective, the pricing model should be established to address the interactions among different brokers. Even thought it is clear that both SPs and users would benefit from this interoperable, federated model, it is not clear what is brought to the identity broker's table other than additional technological costs and legal overheads.

In the work of Zouari and Hamdi [17], an interesting approach to the problem of interoperability is proposed and named An Identity As A Service Framework (AIDF). The AIDF framework relies on the two types of accounts; one for IPs and one for SPs. The framework acts as a matchmaker between the two by helping SPs find a suitable IPs based on their requirements. The IPs list out the type of identities they handle which helps the SPs make an informed choice. The aim of this service is to enable SPs to work with multiple IPs that exist out there in a streamlined manner. The service makes it easier to do so with the feature of single sign in that allows SPs to log in and access all relevant IPs with a single sign in, instead of having to sign in individually for each of the IPs. The service also provides claim transformations for the following standards: SAML, OpenID Connect, Oauth and WS federation. In essence, claims in one standard can be converted into a claim of another standard to match the requirements of another IP and thereby ensuring interoperability. While this framework boosts interoperability, we argue that it has some loopholes that are concerting. Presence of malicious service providers in the system could endanger the contents shared by the IPs. It is thus an imperative to provide some checks for enitites that can register as a SP. Use of secured channels must also be ensured to facilitate safe transport of data. Lastly, the IPs should proactively scan for requests that concern personal information and should have strong policies and guidelines for ensuring that only the necessary data is sent back to the SP.

8 Visions for the Future

The fragmentation of identities is becoming a serious issue. User identities are distributed over different SPs, they are represented in a way that is not known to users and they are no longer in control of their data. In the work of Ates et al. [9], it is stressed that the fragmentation problem is the direct consequence of the current architecture of the Internet. The users are at the periphery, accessing online services through their browsers, while they should be incorporated inside the architecture itself. The authors point out that this issue cannot be resolved by patching the current Internet architecture - the overall change should take place. They propose the total separation between the storage of identity data from their exploitation by SPs. In the proposed Identity-Centric Internet architecture, the entities that would act as a mediators in communication between users and their identities on one side and SPs on the other are called Cloud Agents (CA). The SPs should communicate with personal CA of a user in order to exploit their outsourced personal data for authentication and authorisation, but only with the user's prior consent. The users can access their CA and be in complete control, any time, of how their personal identity data is represented and used by other parties. The identities will no longer be fragmented among different SPs. The main challenges in this architecture are: where the outsourced data should be stored and where the CAs should be hosted. Outsourcing the identity information implies great security threats, while CAs represent the single points of failure. The authors argue that CAs should be replicated to be more resilient, but this does not protect users from unauthorised accesses to their CA. Furthermore, CAs should be highly available at any time, since the communication between users and SPs is not possible without them. Another issue is the integration of CAs - how should they be found and reached by SPs in order to access user identities. This architecture would resolve the fragmentation problem, but besides great security and privacy threats, it would completely change the way we use Internet today. This shift would require great efforts, especially at SPs' side, and should be performed gradually. It would also entail enormous legislation efforts on a global level for the complete and secure adoption.

An extension of the Identity-Centric Internet has been demonstrated in the work of Dash et al. [10], where the authors propose a mobile application as a centralized Cloud Agent that handles the identity management by integrating proxy re-encryption into the widely accepted OpenID Connect protocol. While the benefits and uses cases are numerous, akin to that of creating a digital passport that could be used to log into various government and non-government services, it is still largely unclear how the underlying concerns regarding CAs (as discussed above) would be addressed. As such concerns are still commonplace, it is yet not suitable for mass usage.

Another core application area in not so distant future for IDaaS could be in Quantum Computing. As Quantum Computing is slowly beginning to re-imagine computers as we know it, a lot of the security measures currently in place could be rendered useless due to the computing power and ability that comes with Quatum Computers. New measures have to be developed to keep up with this development. This would require a high amount of expertise in the field of both Quantum Computing and Security Systems (which can be thought of to be a part of IDaaS). In the work of Li et al. [18], an approach that works for Quantum Computing is suggested and it could be later modified to fit more common applications that arise of it. The authors discuss two QI-BQCs (Quantum Identification in Blind Quantum Computing), one based on a single server and the other on a double server. This improves data integrity and identity verification to a great extent and avoids man-in-the middle or DoS (Denial of Service) attacks by eliminating unjammable public channels (channels in which data can be read but not altered) between various entities. However, these protocols cannot be extended to certain simpler applications, even in the realm of quantum computing, because of the aforementioned removal of unjammable channels. They might be necessary in few cases (e.g. in communication between an entity and a Third Trusted Party) that lack other alternatives. This is a question that is outside the realm and scope of our expertise, but could be looked at with great interest for future developments.

9 Conclusion

In this review, we aimed to explored the concept of IDaaS, the main challenges in this field and the so far solutions proposed solutions to address them. The review was intended to familiarize those who consider outsourcing their IMS in the cloud with the challenges that the adoption imposes, but also help IPs in addressing these challenges in their IDaaS. All identified challenges stem from the issue

of trust - the costumers shall trust the third party to manage their identities in the way it is agreed. Feasibility of the proposed solutions is explored from both technological and economic aspects. We conclude that there is no solution that addresses all identified challenges and most likely it will never exist. Some of the proposed solutions imply certain constraints and they are applicable only in certain settings, while others require trade-offs among different challenges.

Solutions proposed to address privacy issues (or more precisely, data confidentiality issues) are based on the proxy re-encryption scheme. The solution in general setting does not seem feasible due to the lack of cryptographic support on user's side. The IPs that want to incorporate this feature and address data-confidentiality should focus on more specific settings. In the realm of eGovernment, data confidentiality is very important and it can benefit a lot from proxy re-encryption feature. Moreover, federated models with pre-established trust on the organisational levels are also perfect candidates for privacy solution. Data confidentiality as an added feature in IDaaS would not require substantial added costs for IPs, but it would require changes at both user's and SP's side which is why it is not suitable for general scenarios. The proposed solution for dependability is based on the replicas of IDaaS components and which should be deployed among different data centres. However, this affects the overall performance of IDaaS and IPs should prioritize which is more important. Telco cloud can be interesting option for leveraging both challenges. The deployment within the infrastructure of an IP is economically impractical for small and medium-sized companies, even though the public cloud deployment imposes privacy issues. Finally, the interoperability issues can be resolved with federated approaches, but they require co-operation among different IPs. If IDaaS wants to be a part of such federation, it gains interoperability as an added value to attract more customers, but it also entails legal and pricing issues that need to addressed.

There seems to be no consensus on which functionalities IDaaS should provide. This can be the double-edged sword for IPs since it gives them flexibility of choice yet it also can make their IDaaS less appealing to customers if they do not provide enough to address their needs. We classified the proposed solutions into the three groups of challenges that they address, yet many other challenges are left untackled and it can be a basis for future works in this field. Even in the realm of privacy issues, most of the solutions aim to address data confidentiality as one subset of privacy. The data privacy during the communication between different entities should be also investigated in some future works. The fragmentation of user identities is yet another issue that emerged. The adoption of proposed solutions for Identity-Centric internet and digital passports does not seem to possible in the near future and there should be more research to investigate the feasibility of these solutions. Finally, Quatum Computers recently emerged as another way that can help to solve fragmentation problem and it will be interesting to see how will this idea be refined in the future.

References

[1] D. Nuñez and I. Agudo, "BlindIdM: A privacy-preserving approach for identity management as a service," *International Journal of Information Security*, vol. 13, no. 2, pp. 199–215, Jun. 2014.

[2] N. Mpofu and W. J. V. Staden, "A survey of trust issues constraining the growth of Identity Management-as-a-Service(IdMaaS)," *2014 Information Security for South Africa*, 2014.

[3] B. Zwattendorfer, K. Stranacher, and A. Tauber, "Towards a Federated Identity as a Service Model," *Technology-Enabled Innovation for Democracy, Government and Governance Lecture Notes in Computer Science*, pp. 43–57, 2013.

[4] C. Emig, F. Brandt, S. Kreuzer, and S. Abeck, "Identity as a Service – Towards a Service-Oriented Identity Management Architecture," *Dependable and Adaptable Networks and Services Lecture Notes in Computer Science*, pp. 1–8, 2007.

[5] A. Gopalakrishnan, "Cloud Computing Identity Management," *SETLabs Briefings*, vol. 7, no. 7, pp. 45-55, 2009.

[6] D. Nuñez, I. Agudo, and J. Lopez, "Integrating OpenID with proxy re-encryption to enhance privacy in cloud-based identity services," *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012.

[7] D. Slamanig, K. Stranacher, and B. Zwattendorfer, "User-centric identity as a service-architecture for eIDs with selective attribute disclosure," *Proceedings of the 19th ACM symposium on Access control models and technologies - SACMAT 14*, 2014.

[8] D. Kreutz and E. Feitosa, "Identity Providers-as-a-Service built as Cloud-of-Clouds: challenges and opportunities," *Position Papers of the 2014 Federated Conference on Computer Science and Information Systems*, 2014.

[9] M. Ates, S. Ravet, A. M. Ahmat, and J. Fayolle, "An Identity-Centric Internet: Identity in the Cloud, Identity as a Service and Other Delights," *2011 Sixth International Conference on Availability, Reliability and Security*, 2011.

[10] P. Dash, C. Rabensteiner, F. Horandner and S. Roth, "Towards Privacy-Preserving and User-Centric Identity Management as a Service," *Open Identity 2017, Lecture Notes in Informatics (LNI), Gesellschaft fur Informatik, Bonn*, 2017.

[11] I. Gomaa, E. Abd-ElRahman, E. Saad, and A. Ksentini, "Virtual Identity Performance Evaluations of Anonymous Authentication in IDaaS Framework," *Egyptian National Telecommunication 2019 Institute (NTI), Cairo, Egypt*, 2019.

[12] T. H. Vo, W. Fuhrmann, and K.-P. Fischer-Hellmann, "Privacy-preserving user identity in Identity-as-a-Service," *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2018.

[13] T. H. Vo, W. Fuhrmann, K.-P. Fischer-Hellmann, and S. Furnell, "An Adaptive Security Infrastructure and Privacy-Preserving User Identity for the Cloud Environment," *Future Internet 2019*, 2019.

[14] T. H. Vo, W. Fuhrmann, and K.-P. Fischer-Hellmann, "Identity-as-a-Service (IDaaS): a Missing Gap for Moving Enterprise Applications in Inter-Cloud," *Eleventh International Network Conference (INC 2016)*, 2016.

[15] N. Mpofu and W. J. C. V. Staden, "Evaluating the Severity of Trust to Identity-Management-as-a-Service," *2017 Information Security for South Africa (ISSA)*, 2017.

[16] T. H. Vo, W. Fuhrmann, K. -P. Fischer-Hellmann, and S. Furnell, "Automated trust negotiation for Cloud applications in Identity-as-a-Service," *2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*, 2019.

[17] J. Zouari, and M. Hamdi, "AIDF: An Identity As A Service Framework For The Cloud," *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, 2016.

[18] Q. Li, Z. Li, W. H. Chan, S. Zhang, and C. Liu, "Blind quantum computation with identity authentication," *Physics Letters A*, vol. 382, no. 14, pp. 938–941, 2018.

Roles and Responsibilities	Name of Member
Introduction	Stijn Veken
Identity Management Systems	Stijn Veken
Identity Models	Milica Đorđević
Early Motivation for IDaaS	Sagnik Aditya
Benefits of IDaaS	Milica Đorđević
Challenges in IDaaS	Milica Đorđević
Solutions	Sagnik Aditya
Visions for future	Sagnik Aditya
Conclusion	Sagnik Aditya, Stijn Veken and Milica Đorđević
Presentation	Milica Đorđević

Intelligence at the edge: A review of Machine Learning in edge computing

Jetske Beks
University of Amsterdam
11065249

Willemijn Beks
University of Amsterdam
10775110

Mike Schouw
University of Amsterdam
11268751

Abstract

In recent years the emergence of big data at the edge, brought on by the dramatic increase in IoT and mobile devices, has given rise to a new field of research into edge intelligence applications. Many of such applications use deep neural networks and are traditionally run in the cloud. Recent research looks into the possibilities of moving these applications to the network edge. In this review paper, we explore in which cases it is valuable to combine machine learning applications with edge computing in this way. We discuss when this development can be appropriate within the framework of latency, scalability, reliability, and privacy/security. We find that the combination of machine learning solutions and edge computing infrastructure is often effective when using distributed models or hybrid architectures. However, there is still a lot of research to be done to optimize the combination between the two fields for useful applications.

1 Introduction

In recent years there has been an increase in mobile and Internet of Things (IoT) devices worldwide. The immense amount of data being generated by these devices, combined with a need to analyse this data, gives rise to a large-scale shift of data storage and computing from cloud data centers to the network edge (Zhou et al., 2019; Wang et al., 2020). Artificial Intelligence (AI), especially deep learning, is a useful tool in processing all this data and extracting information from it. Therefore, it is often used for doing computation at the edge. The combination of these paradigms is called *edge intelligence* (Zhou et al., 2019).

To talk about edge intelligence, we will first define what *edge* is. Shi et al. (2016) describe it as as any computing or network source between the path of a data source, also called an *end device*, and cloud data centers (Shi et al., 2016). When using edge devices to perform computation, usually on data that is generated in close proximity of the device, this is called *edge computing* (Xu et al., 2019). As the field of edge computing has been emerging only in the last five years or so, there is no consensus yet reached on exactly which paradigms and technologies fall within its scope. Some studies exclude end devices from their definition of edge computing (Lopez et al., 2015); some studies use the term *fog computing*, which seems to be similar to edge computing but not quite the same (Matt, 2018); and some use other terms altogether, which are sometimes interchangeable (Dolui and Datta, 2017). In this paper, we will be using the definition from Wang et al., who specify edge computing as encompassing all technologies for which some amount of computation is done at edge or end devices.

The use of AI in edge computing has multiple benefits, as there is an exchange between them where both fields improve (Zhou et al., 2019). AI helps edge computing with the copious amounts of data that now exist at the network edge, which need to be handled by AI to extract the most useful information from it (Wang et al., 2020; Dey and Mukherjee, 2018), while edge computing can change and improve AI with new application scenarios (Zhou et al., 2019). Deep learning especially is valuable in combination with IoT devices, since these devices often generate data from a noisy and

Preprint. Under review.

complex environment. Conventional machine learning approaches get confused by these types of data, but deep learning seems to be a promising solution for this (Verma et al., 2017). However, the efficiency of deep learning comes at the price of high computational power and memory requirements (Chen and Ran, 2019). To leverage the power of deep learning, the cloud is often used, which can usually satisfy these requirements. But to utilize cloud resources, the data must be moved from the data source to the cloud infrastructure. This presents various challenges, such as high latency and issues with privacy. Edge computing can be a viable option to tackle these problems. However, doing deep learning on the edge is also not without challenges, such as fulfilling high computational resource requirements on not-so-powerful edge devices (Xu et al., 2019). Storage of data can also be a problem, especially as it keeps building up. Furthermore, there is a need for coordination between edge devices and with the cloud to ensure the best performance (Chen and Ran, 2019).

Taking all of this into account, we find that it is vital to discuss the challenges that arise with new techniques like this among with the possible benefits. The question that we are trying to answer in this work is, therefore, *in which cases is it valuable to combine machine learning applications with edge computing?* Our hope is that the themes we identify within the existing literature will help to better understand the current state of the field, and push it forward in directions that will be worthwhile for both researchers and users of edge intelligence.

To better understand the rest of this paper, we will provide in Section 2 a theoretical background on relevant machine learning theory. Then, we will identify some influential factors in the context of edge intelligence using recent literature. These are summarized in Section 3. With the help of these factors, we will examine current applications of machine learning in the network edge in Section 4, analysing their usefulness. Lastly, Section 5 will discuss our general findings and present our conclusions.

2 Machine learning background

When deciding whether to apply machine learning (ML) in edge computing applications, it is imperative to understand what exactly the possibilities are that ML offers, why they could work, and what the constraints are. For this reason we will now discuss the relevant parts of ML in a high-level way, so that the discussion about different applications can be understood sufficiently.

ML can be understood as having three main categories, which are supervised learning, unsupervised learning, and reinforcement learning. With unsupervised learning, a model is created that finds a structure in training data that has not been labelled. Examples of this technique would be something like a network learning to differentiate between pictures of cats and dogs, not knowing beforehand which are which. Or a generative adversarial network (GAN) that is used to create new datapoints based on some dataset, which are representative for the set but anonymized. Supervised learning is defined by the training of a network through feeding it a labelled dataset, and the model learning to perform either regression or classification on it. Finally, reinforcement learning works by defining an agent and a world, with rules for the agent to prosper in that world. By having iterations of the agent taking actions in the world, the agent receives feedback and uses this to change and improve its actions. Learning the preferred action of some agent is usually done by neural networks, with models where the parameter set is quite large. Reinforcement learning is often used for games where the opponent is managed by an artificial intelligence.

The main ML technique that is used throughout most research in edge computing that we found for this paper are neural networks (NN), and specifically deep neural networks (DNN). To understand how these pertain to ML in general we will explain briefly what the different techniques are within this field.
A neural network is an architecture consisting of multiple layers of nodes that connect in some sense to the previous and next layer. The first layer is the input layer, the last layer is the output layer, and all possible layers in between are the hidden layers. A neural network with at least one hidden layer is called a *deep* neural network. The nodes calculate their values based on the weighted sum of the previous nodes that are connected to it. To get a non-linear model, an activation function is used. This process repeats at every layer and finally, the output is calculated at the output layer (Chen and Ran, 2019).
The training of such a network happens with stochastic gradient descent (SGD), which applies many such iterations. First a forward pass through the network is done with a random sample. Then every

time an output value is calculated, the result is compared to a known value (supervised) or some other result measure (unsupervised), and the error is back-propagated through the network and the weights at each node are updated. This process starts at the final layer of the model and ends at the starting layer (Chen and Ran, 2019). For each pass, or epoch, a randomly selected 'mini-batch' of data is used to update the gradients so that the training loss is minimized (Ruder, 2016).
Training a network can be done for any preferred amount of iterations. Nonetheless, it is possible to train too little or too much, which can lead to underfitting or overfitting respectively. Underfitting refers to a model that does not capture the subtleties of a dataset, while overfitting refers to a model that also captured the outliers within the test set and thus does not generalise well.
When the training phase is done, the result for a single (new) datapoint can be obtained by entering it in the trained network; this process is called inference. The type of result depends on whether the dataset that the NN was trained on modelled regression or classification. There are many different types of NNs that all have their own domain of application, e.g. convolution neural networks (CNN) which are used for image processing, or recurrent neural networks (RNN) used for time-series problems.
To understand how the current challenges in distributed ML techniques came to exist, it is crucial to know how the current ML applications and research are mostly developed. Until recently, it was assumed that ML applications would run centralized and offline (Park et al., 2019). This meant that hardware was assumed to be sufficient, which is a different situation from where we are now in terms of research. Investigations into edge intelligence change the focus towards lower capacity hardware and wireless communication, introducing with this development some new challenges. A prominent part of these new challenges arises while training a ML model. Training is one of the most expensive parts of a NN in terms of memory, energy, and time. And since distributed training hinders convergence of the model, this is an important problem (Park et al., 2019). When training a ML model, a lot of communication happens between each layer, going in both directions due to back-propagation. To make it feasible to train somewhere else than in the cloud alone, there are some techniques to facilitate this. Training can be architecturally split, by either splitting the data or the model; or devices can exchange parameters by using a centralized parallel SGD or an elastic SGD (Park et al., 2019). There are multiple ways to apply these principles. One such approach, which is quite successful, is federated learning (Zhou et al., 2019). This approach uses a shared model that is updated with intermediate results from multiple sources, ensuring privacy by keeping the data itself on the devices. Likewise, there are more approaches to splitting the training phase or the inference calculations, which all have different primary focus points. Such focus points are the themes that will be discussed in the next section.

3 Themes

To discuss the value of different applications of deep learning on edge devices, first some important factors influencing it are introduced. These factors are used in existing literature to justify moving applications from cloud to edge environments, but are also considered, often but not always in a different form, problems in the edge computing context itself. We will discuss these factors from the perspective of both: these are advantages and disadvantages to consider when deciding where to place a deep learning application.

Latency The factor that is most often named in edge intelligence application papers is latency, which is connected to reliability and scalability of the network. If the network is slow, the distance between the central cloud and the edge device can be crucial: some time-critical applications will have strict delay requirements that cannot be satisfied with normal cloud computing (Wang et al., 2020; Zhou et al., 2018). According to Chen and Ran (2019), there can also be additional queuing in the cloud infrastructure on top of propagation delays, resulting in an even longer end-to-end time.
With the increasing amounts of 5G and IoT connectivity, latency can rise even further with the number of devices connected to the network (Xu et al., 2019). However, 5G could also provide solutions for the latency problems with increasing speeds, increasing bandwidth and decreasing network latency (Ren et al., 2020).

Latency between edge devices is usually not an issue, as these tend to be a lot closer together physically. However, when distributing models over different devices, the extra communication and data sharing needed could also introduce undesirable latency (Chen and Ran, 2019; Matt, 2018)

Scalability Scalability is closely related with latency, and sometimes used interchangeably in the literature, as network bandwidth seems to be the main problem in the edge intelligence field. Chen and Ran (2019) and Xu et al. (2019) cite network access to the cloud as being the bottleneck for deep learning applications, especially if the cloud server is in a central location. Shi et al. (2016) adds that a lot of edge devices in one area might challenge current network bandwidth. Lopez et al. (2015) consider also the storage costs of the data generated by edge devices. The total amount could be up to 850 ZB in 2021, compared to 20 ZB generated by data centers (Zhou et al., 2019). Aggregated data from an abundance of sources could overwhelm even big data centers. Not only storage capacity of the cloud could be problematic, but also the associated increase in processing time, especially when limited finances need to be considered (Lopez et al., 2015; Wang et al., 2020).

On the other hand, scalability is just as complicated in edge devices, if not more so. End devices such as IoT sensors often have low processing power and storage capabilities (Wang et al., 2020; Qi and Liu, 2018). On top of that, their energy consumption is often a bottleneck, especially in mobile devices with small batteries. Running complex applications on these devices quickly depletes their energy reserves and can lead to dropout (Shi et al., 2016; Li et al., 2018a). Edge servers are not quite so energy constrained and usually have higher capabilities for computing and caching, but depending on the type of hardware this could still be prohibitive for deep learning models with large numbers of parameters and layers (Wang et al., 2020).

Reliability In some cases, constant availability of the application is very important, such as for health care, autonomous vehicles, or inside factories. Most cloud connections rely on existing network structure that they have no control over, which is not always reliable enough for these applications Wang et al. (2020). Cloud outages, which can be caused by power outages or system flaws but also denial of service attacks, are also cited as a major argument for moving deep learning applications to the edge (Dey and Mukherjee, 2018; Lopez et al., 2015).

However, edge devices can also be very unreliable due to connection dropout or even node crashes. Detecting and identifying the reason for these failures could be problematic (Shi et al., 2016). The mobility of some edge devices is also an issue. Network connectivity varies greatly over regions, and devices moving too quickly could cause problems when running distributed models. One solution for this could be to keep all computation on-device (Dey and Mukherjee, 2018).

Privacy & Security Lastly, there are some ethical issues with the centralized aggregating and storing of personal data in clouds (Lopez et al., 2015). Not only could an attacker get data from millions of users at once, some users also don't trust the cloud provider to handle their data well. Privacy is therefore a significant concern associated with clouds. Especially data required for learning applications might contain a lot of private information, and applications for smart homes and the like are very vulnerable (Wang et al., 2020). According to Shi et al. and Li et al., using edge computing would improve privacy in the way that sensitive data does not need to be processed in a centralized data store anymore. By keeping the sensitive data local, and only reporting the inference outputs, edge computing could be a decent method to protect privacy and data security.

However, because edge devices are often so resource-scarce, using some of these resources for security applications could drastically downgrade their capabilities for machine learning, and current existing tools might not work or not work as well for these devices (Lopez et al., 2015; Shi et al., 2016). There is also the problem of user incompetence or unawareness, for example when not updating default passwords on end devices (Shi et al., 2016).

When using distributed models, including edge nodes or end devices that are not controlled by you could also carry the risk of security flaws, and some could even be malicious (Lopez et al., 2015). A solution for that could be a hybrid approach, where parameters or other parts of the model that do not directly contain any sensitive information are the parts that are communicated, while the raw data remains on the end devices (Park et al., 2019; Wang et al., 2020). As was mentioned in the ML background section, there are multiple approaches to ensure privacy when training in a distributed way.

4 Applications

We will now discuss some of the current applications of deep learning in edge, which have various goals and come from multiple fields. Using the factors covered in the last section, we will attempt to compare them and evaluate whether the combination of deep learning and edge computing was actually a valuable one, or whether the cloud might after all be the better option for some applications.

4.1 Image Recognition

An application that is often moved to the edge is image recognition, used for a variety of purposes. This is usually because the size of the image or video data needed for training a model such as CNN grows very large very quickly. And of course large image data is also needed for inference. The shorter the distance all this data needs to travel, the better.

Liu et al. (2018) have developed deep learning based food recognition software using edge computing. One of their aims was to design a system that can minimize response time and energy consumption. This was done by splitting the food recognition tasks between edge devices, specifically smartphones in their proposed use case, and the cloud. Their proposed structure first lets the edge device perform lightweight image preprocessing tasks, after which the transformed image is transported to the cloud server, where the heavy computation of deep learning model inference is executed. Their results, when compared to existing systems, outperform in accuracy and achieve roughly the minimum response time and energy consumption of the state-of-the-art.
To judge how their edge computing infrastructure improves the results, Liu et al. test their system against two others with a different infrastructure. One runs computer vision algorithms for food recognition on the edge device itself, without sending anything to the cloud. The other just sends the images to the cloud without preprocessing them, and runs a different deep learning model there. Their proposed infrastructure seems to perform better than these two others. However, these systems are not good baseline systems to test against. For the first system, it is understandable that in order to be able to actually compute everything on a resource-scarce edge device, less computation-heavy models are used. Unfortunately, this is not explained and there is no argument made for why these model types are the most comparable models in this context. And the second system actually tests three things: the influence of the deep learning model, the influence of doing data reduction on the edge device (by way of the preprocessing steps), and the influence of using the preprocessing steps at all. These things could easily have been disentangled by using more testing systems with one difference each in infrastructure or model, instead of one completely different system. They also did not include energy consumption at the server in the measurements, which obviously does not give a fair comparison. This all means that we do not actually know the influence of using edge computing here, even though it seems theoretically useful in terms of possible latency and even privacy improvements.

Huang et al. (2018) have developed a mosquito classification system, using a self-built device including a Raspberry Pi that attracts mosquitoes in the area and photographs them. Edge computing is used on the device to reduce the size and do some preprocessing on the images before they are sent to a data center. There, further preprocessing is done and a CNN model classifies the images into mosquito types.
Huang et al. have tested their system in the wild, where it achieves an accuracy score of 90%. However, there is no comparison done with a non edge computing implementation, so it is not clear how much edge computing benefited the system. They do mention that during the development phase, *all* preprocessing tasks were done on the device, but this made it too slow to keep filming the mosquitoes. No argument is made for why any computing should be done on the device at all, and no time or energy related measures were taken. We theorize that identifying mosquitoes real-time, so that they can be killed or caught immediately if they are of a certain type, could be a valuable function of such a device. Nevertheless, this idea is not apparent from the paper, and more research should be done regarding the edge computing infrastructure before it is workable.

Yang et al. (2020) use deep learning to identify different types of crops growing on a field. They train two models, SegNet and a version of AlexNet, to differentiate crops based on images. This training phase is done in the cloud. The resulting model is then put on an edge server. A drone is flown over a field and takes images of it, which it immediately sends to the edge server over the 4G network. The server runs the received images through both models, and the inference results are used to adjust the route of the drone by sending a signal to the controller. How exactly the classification into crop types

results in a route adjustment for the drone is not explained; we assume that images capturing a road, which is also a type of "crop" that can be classified, will cause the drone to be turned back towards the field in some way.
The accuracy of the classification is about 90%, which is promising. The inference on the remote server takes less than a second; however, there is still a network latency of 3 to 4 seconds. Yang et al. note this but say that this latency was not the subject of their study and that possibly 5G will fix the delay.
Since we don't know the route adjustment process, it is not clear whether this delay is limiting for the drone. However, another constraint that *is* mentioned is the battery time: using a single battery, the drone can only fly for about 10-13 minutes. This will not be enough for the use cases that Yang et al. propose, which include possibly classifying crops over the whole of Taiwan. Nevertheless, the idea is useful and with some more research on latency and energy consumption of the drone, it could perform quite well.

Summarizing, we see that image recognition systems using edge intelligence employ hybrid architectures to deal with the high computing power needed by the deep learning models. Image preprocessing can be done on the end device, but model training and sometimes also inference is still done in the cloud. Preprocessing on the end device does reduce the size of the data that is sent on, which proves useful. We see that filming also uses up a lot of power. Experimenting with better hardware in the end devices could be useful in these cases.

4.2 Mobile Cloud Sensing

Mobile cloud sensing (MCS) systems use the fact that mobile phones have a lot of interesting sensors that can be used to measure a variety of things. Most MCS systems rely on a central server which processes and validates all the data collected from the users before analysing it.

Zhou et al. (2018) have developed a robust mobile cloud sensing (RMCS) system with deep learning and edge computing. The proposed RMCS system attempts to leverage edge servers for a decrease of network latency to the central server. Deep learning is implemented at the edge nodes to preprocess and validate raw data such as images and video, before sending them to the cloud for analysis.
When using the RMCS system for a case study with smart transportation, the system was indeed able to filter out irrelevant, redundant and invalid data at the edge servers. The results show increased robustness – the percentage of useful data in the total collected data – up to 94% from 72%. By doing data validation in the edge servers, Zhou et al. also reduce the amount of data sent on to the cloud by about 75% in their case study. This indicates that using edge computing for this purpose will probably reduce latency and network overload quite a lot, even though these things are not measured separately. The links leading from the mobile devices to edge servers are still burdened with the full dataset, but the authors cite low energy efficiency of the deep learning model as being prohibitive for implementing it directly on the mobile device. They do not give any specifications or measurements on this, however, so maybe more research into this could prove how prohibitive it really is. They also mention security threats for mobile devices as an open issue. Security issues resulting from malicious devices sending fake data, however, are resolved quite well already by including a data authentication step in the deep learning model.

4.3 Security

Edge intelligence can also be used for advancing security in the edge. Tian et al. (2020) has developed a deep learning system to detect web attacks by use of well-designed URLs. They use multiple collaborating models for this, including word2vec and CNN, which would run concurrently on different servers. The models process the URLs and check them for anomalous patterns and keywords.
Tian et al. test their system with two concurrent models on a device which is not particularly resource-constrained, and get quite high accuracy scores, which they claim are competitive with other approaches to this problem. However, it must be noted that they do not test the system in a real-life edge architecture, and while they cite the goal of the system as being for use in edge devices, there is no discussion about architectural decisions for the models being made in consideration of this fact. One would expect that models running on edge servers would need to have lower processing costs, as they need to run continuously in the background for it to have any effect on the security, and no mention is being made about communication between models running on different servers.

Such a detection system for web attacks does seem useful in the context of edge servers. Although they are less attractive for attackers because of the small amount of users for whom data can be found in an edge server, compared to a cloud database it will not be secured as well, as described in the previous section. Applying deep learning is also useful, as prove the results: the system is very accurate, more accurate than systems not using deep learning but, for example, regex-based approaches. The question is though how well it will really perform in an actual edge infrastructure, instead of the experimental environment with only one server they use.

4.4 Healthcare

Healthcare is a field in which it is especially important that data is kept private, and in some cases also that applications run real-time. In the case of Hosseini et al. (2017), it is both: they use deep learning for monitoring, evaluation and regulation of epilepsy. They create multiple models with different purposes: estimating the location of epilepsy in the brain, using this to decide on a treatment, and detecting oncoming seizures in the brain. The idea is that edge computing is used in an edge layer directly connected to biosensors. It would preprocess the large amounts of sensor data before sending it on to the cloud for further processing and storage, as well as monitor and provide alerts or stimulation mechanisms based on this data.
Hosseini et al. cite the cost of scalability, privacy issues, and latency as reasons for their choice of edge computing. Patient health data is better managed and shared locally for privacy reasons, they argue, and alert systems need to be as low-latency and reliable as possible for the safety of the patient. This latency is tested by sending the experimental dataset to both the remote cloud and to the edge devices and comparing the round trip times. They conclude that the latency between sensor to edge devices is about 10 times smaller. However, except for this isolated field test, all models are tested on small, curated datasets and in controlled environments. So not only are most of the results theoretical, the value of using edge devices for this is also yet theoretical. Further research should build on this.

4.5 Smart City

Using IoT devices and other technologies to manage urban areas could optimize the running of city services, with less manual labor needed to aggregate information and transform it into useful formats. This concept is called ‘smart city’. In this context, there are a lot of potentially valuable edge computing applications.

Zhang et al. (2019) implement a ‘urban street cleanliness assessment model’, the goal of which is to detect garbage on city streets automatically so it can be cleaned up in a timely manner. They cite the high latency of central cloud as being the relevant factor for deciding to use edge computing for this problem. Mobile end devices, such as garbage collection vehicles and smartphones of citizens, are used for taking photos of city streets. This material is sent to edge servers, which are installed all over the city. The edge servers contain stationary units but also units installed on buses, which are mobile. The image data is preprocessed there, mostly by doing filtering on road areas and adjusting image size. The data is then sent to the central cloud for the garbage detection step, where both training and inference are done.
Zhang et al. explicitly test the performance improvement of using distributed edge computing for preprocessing over immediately sending all data to the cloud. This improvement is quite large; they do not discuss this further, but it seems that the time decreases linearly with the amount of devices doing the preprocessing. However, it is unclear how exactly the preprocessing is done on the edge servers, especially as these seem to be only NAS units, which usually do not run software other than what is needed for data storage. There is no algorithm given for the image size adjustment. A mention is made of having humans manually overlook the material for filtering on roads, but they also explicitly say that the edge servers handle this step, although no word on how. As it is not clear what exactly the preprocessing step accomplishes, the performance improvement does unfortunately not mean anything to us. The hypothesis is sound: using multiple distributed edge servers to preprocess the image data could indeed improve the end-to-end performance, since the cloud and the network leading to it need to do less work on less data. This would be a latency and scalability improvement. However, the question can be asked why it is necessary to do this cleanliness assessment in ‘real-time’, as they desire, since garbage cleaning is not performed constantly throughout the day.

4.6 Intelligent Edge

A field of research that has a lot of overlap with edge intelligence is *intelligent edge*, as Wang et al. summarize it. This consists of the application of ML to optimize the performance of edge computing in various ways. Intelligent ways to split computations between edge and cloud, distribute models over edge devices, and detect failure preventively are useful tools and can help other 'basic' edge intelligence applications to run better and more effectively. We will examine multiple studies proposing such methods for optimizing edge computing.

Kang et al. (2017) have developed a technique for intelligently dividing a DNN between the cloud and the network edge, called Neurosurgeon. According to the authors, Neurosurgeon is a lightweight scheduler that partitions DNN between the cloud and the network edge, at the granularity of network layers. Their reasoning is that data transfer is becoming a bottleneck for applications on edge devices, while energy efficiency is increasing in these devices. Consequently, the main questions Kang et al. answer in this paper are about feasibility of running large-scale intelligent workloads on a mobile platform, figuring out when the effort of sending large quantities of data through the network for efficient cloud computing is the better option, and defining the role of mobile edge in ML applications that require heavy computation.
Kang et al. make an explicit distinction between the different types of layers in the distinct types of DNN. This is essential for the functioning of Neurosurgeon, since the different types of layers behave very differently in terms of amounts of input and output data in terms of quantity. This research also implicitly focuses on execution of the network and not on the training phase, which might warrant future research. The key observations made by Kang et al. about the behaviour of different networks are generalized, without theoretical arguments, which should lead to additional research. Kang et al. find that their scheduler achieves an average latency speedup of 3.1x, with a best case speedup of 40.7x, has energy reductions between 59.5% and 94.7%, and a throughput improvement of 1.5x to 6.7x. However, on closer inspection it appears that almost all performances that differ significantly from the cloud-only approach in terms of latency and energy consumption are accomplished with the same DNN (Senna, which consists of three layers). Neurosurgeon places the computation completely at the mobile device in all cases. This is not commented on by Kang et al.. They also do not comment on the possible overhead of neurosurgeon itself, as they have not measured it. They do explain that it is very fast since it consists of regression functions with different parameters based on the different DNN layer types. However, in some cases there is no noticeable improvement in performance from using Neurosurgeon, which might make it worse than the cloud-only approach in some cases. Neurosurgeon is very useful in some cases like the word-vector DNN (Senna) that was found to have significantly improved performance. More research should be done on exactly when Neurosurgeon is useful. Kang et al. have given an insight into how a hybrid approach between the cloud and the network edge is possible and might sometimes be an optimal solution. However the training phase of the DNN was excluded, which simplified the problem at hand.

Li et al. (2018b) also presented a scheduling solution, in this case for optimizing deep learning for IoT in combination with edge computing. In the paper it is mentioned that its main contributions are defined by introduction of a new elastic model for deep learning methods for IoT into edge computing, which is claimed to be an innovative application. Li et al. explicitly exclude IoT devices from their definition of edge computing, since they often have lower capacity chips and will not be able to process many deep learning calculations. Their NN is trained in the cloud, which does not diverge from the traditional approach. The focus in performance increase is on the execution instead. The choice to not include training of the DNN in their research significantly simplifies it, considering they do not have to include the latency of back propagation communications going from the cloud to the network edge while training. This means that there is potentially significantly less traffic to deal with, thus simplifying their problem to a scheduling problem. The main issue becomes finding the right balance between using powerful, but slower to reach cloud computing, or less powerful but faster to reach edge computing. They use an offline algorithm that schedules layers of DNNs over different edge servers, and an online algorithm that actually deploys them, so that they are distributed in an optimal way considering the difference in available resources of the servers.
Li et al. distinguish themselves from other research by explicitly excluding mobile devices from their research, enabling them to find results on how efficient it is to use only the network edge in combination with cloud. They lack a deeper analysis of their results by not making a sufficient distinction in different types of DNN, only considering a few CNNs, as these could impact the ratio between cloud and edge computation differently. They also test their scheduler only in a controlled

environment that simulates the different servers, and not in a real-world experiment. The positive results do imply that with more research there could definitely be more merit in their approach. Li et al. conclude that they showed that their scheduler is able to increase the number of tasks that can be run on edge servers while maintaining a QoS requirement.

With the introduction of 5G to the world, the scope of coverage of IoT could be increased more easily. This means that the attraction of doing edge intelligence with IoT grows, resulting also in the increase of computing power needed for mobile applications (Xu et al., 2019; Taleb et al., 2017). In order to avoid overloading the edge computing infrastructure and properly balance the offloading between cloud and edge computing infrastructure, Xu et al. (2019) propose an Heuristic Offloading Method (HOM) for deep learning edge services in 5G networks. The model is based on a centralized unit versus distributed unit architecture. This architecture connects multiple mobile devices to a distributed unit, distributed units to centralized units, and these are connected to the cloud. The goal is to offload 'deep learning tasks' from the mobile devices to either a unit server or the cloud, minimizing the end-to-end time. Tasks are first moved to the centralized unit that is closest to a device, which then decides whether to execute the task itself or schedule it to either another unit or the cloud. These tasks seem to be at a bigger granularity than what we've seen before; however, Xu et al. are not very precise as to what these deep learning tasks consist of exactly, which is unfortunate. The results of Xu et al. indicate that their HOM method has the lowest latency in all their testing methods when compared to Cloud Offloading (CO), where everything is offloaded to the cloud, and Local Offloading (LO), when everything is offloaded to the closest edge server. They, too, use simulations to test their method, so it is possible that this result does not transfer to real 5G networks, but it is logical since these networks are not yet widely available to test on.

Another 5G offloading approach is done by Ren et al. (2020). They propose a fine-grained collaborative solution for mobile AR, which partitions DNNs over cloud, edge, and mobile web browser. More specifically, the offloading approach moves the high performance computation to the cloud infrastructure, while the rest is placed on the user device. This will reduce cloud computing costs. They test this collaborative solution on a trial 5G network, comparing it with cloud-based, end-based, and other offloading algorithms. They pose, however, that because of the increase in network speed 5G brings, these other offloading approaches degraded into cloud-based algorithms. The results indicate that their collaborative approach has better latency, saves cloud computing costs, and reduces mobile energy consumption compared with the other approaches. Ren et al. (2020) note that these efforts are still preliminary attempts and much work remains to be done.

Both Xu et al. (2019) and Ren et al. (2020) have developed different offloading strategies with deep learning and edge computing in combination with 5G networks and both show improvement on their performance metrics. However, both offloading methods have not been tested in real life environments. Therefore it is hard to say what the actual performance improvement of these offloading methods will be.

5 Discussion and conclusions

In this work we have attempted to define the value of moving deep learning applications from the cloud towards the edge in terms of scalability, latency, privacy, and reliability. These factors introduce problems at both the cloud and edge ends in different ways, and have considerable influence on the difficulties and benefits of doing deep learning at either end.

Reducing latency is cited in all applications we have reviewed as a main reason to move to the edge, and a lot of the applications indeed see a latency improvement. Additionally, the 5G network could provide even more latency improvements, for both the cloud and the edge side, even though it could need different strategies for offloading. However, there are some applications where the need for lower latency does not seem to be very important to its functioning. In these cases, it is perhaps better to first weigh the disadvantages of moving to the edge; low latency is nice, but maybe not at the cost of privacy or reliability.
Scalability seems to be not so much a problem in the cloud as it is in the edge, with multiple techniques specially designed to get the most out of the low power of edge devices. However, cloud services do often cost money, and the financial benefit of moving to the edge could offset this disadvantage. Hybrid architectures, where the initial model is trained in the cloud but then moved to the edge to be used in applications, seem to be a very promising avenue to take here.

Reliability of the cloud is also not often a real problem for applications, but in the cases that it is, a distributed approach using trusted or self-controlled edge nodes is a good solution. Likewise, for the edge, distribution is often the best solution when worried about the reliability of a single node. Some headway is made already into doing this in an effective manner, and we think that with the maturing of this field, good frameworks for doing this with general ML models will eventually arise.
Lastly, privacy is often a more theoretical issue, but it could have disastrous consequences if it turns out that it wasn't protected well enough. For applications in fields like healthcare, the privacy requirements should weigh heavily in the decision to move to the edge. This works well if you have control over all the edge devices. Hybrid approaches, where raw data is kept close to the user but model parameters or other intermediate results are shared with other edge nodes or even cloud centers, seem again to be a good approach to take here.

We have seen, in general, a lot of hybrid approaches in the literature. It seems that deep learning is still too resource-heavy to move completely to the edge in most cases. Nonetheless, preprocessing the raw data on the edge device to reduce the size, remove irrelevant datapoints, and transform it to a shape that the model can take in, is a very useful step to take. Not only does it reduce latency, it also sends on purely data that is useful, which reduces storage cost for the server that receives it; and edge devices are usually capable enough to do this small amount of processing. In practically all use cases this seems to be worthwhile to do.
Distributing the ML model is also often a valuable approach for multiple reasons. There exist multiple tools already that try to do this in an optimal way, often at the granularity of network layers, although not for all types of DNNs. There is also very little regard for security in these cases. As the work moves from theoretical to real-life use cases, this aspect should be taken into consideration.
Location awareness, which is mentioned in some review papers but never actually used in practice, is something that also should be considered more. It is useful from a machine learning perspective, as location awareness can improve results of models a lot, depending on the application. It is not a challenge that should be solved, but rather a benefit of moving to the edge completely. More research into its effectiveness specifically for the edge could be very valuable.

The choice for edge or cloud for a machine learning application, as with most things, depends a lot on its particular context and needs. Consider whether the application has strong privacy or reliability needs, or has to work in real-time. Take into account the costs of using the cloud, and the amount of (trusted) edge nodes you have access to. In the end, the decision can only be made for a specific application in consideration with its specific needs. Nonetheless, we hope that this paper has given a greater understanding of the current state of the field, and that it has clarified the benefits and costs of moving an application to the edge.

To conclude our research, the combination of machine learning solutions and edge computing infrastructure is a very promising prospect and the current research shows encouraging results. However, there is still a lot of research to be done to optimize the combination between the two fields for useful applications. Furthermore, a lot of applications that have been proposed by research are mainly theoretical and are not deployed in the real world yet. Nonetheless, hybrid solutions between current cloud infrastructure and edge computing infrastructure may be the best solution for the current problems that we are facing with regards to scalability, latency, reliability, privacy and security.

6 Participation

Table 1: Task Division

Name	Tasks
Jetske Beks	• first draft themes • read some of the application papers • wrote summary and evaluation for these papers • second draft conclusion • edit other sections
Willemijn Beks	• second draft introduction • machine learning section • read 2 long review papers referred throughout paper • read 2 application papers and wrote reviews for them • first draft abstract
Mike Schouw	• first draft introduction • read multiple application papers • wrote summary and evaluation for these papers • first draft conclusion

References

Chen, J. and X. Ran (2019). Deep Learning With Edge Computing: A Review. *Proceedings of the IEEE 107*(8), 1655–1674.

Dey, S. and A. Mukherjee (2018). Implementing Deep Learning and Inferencing on Fog and Edge Computing Systems. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2018*, pp. 818–823. IEEE.

Dolui, K. and S. K. Datta (2017). Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. *GIoTS 2017 - Global Internet of Things Summit, Proceedings*.

Hosseini, M. P., T. X. Tran, D. Pompili, K. Elisevich, and H. Soltanian-Zadeh (2017). Deep Learning with Edge Computing for Localization of Epileptogenicity Using Multimodal rs-fMRI and EEG Big Data. In *Proceedings - 2017 IEEE International Conference on Autonomic Computing, ICAC 2017*, pp. 83–92.

Huang, L. P., M. H. Hong, C. H. Luo, S. Mahajan, and L. J. Chen (2018). A Vector Mosquitoes Classification System Based on Edge Computing and Deep Learning. In *Proceedings - 2018 Conference on Technologies and Applications of Artificial Intelligence, TAAI 2018*, pp. 24–27. IEEE.

Kang, Y., J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang (2017). Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News 45*(1), 615–629.

Li, H., K. Ota, and M. Dong (2018a). Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Network 32*(1), 96–101.

Li, H., K. Ota, and M. Dong (2018b). Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network 32*(1), 96–101.

Liu, C., Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou (2018). A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. *IEEE Transactions on Services Computing 11*(2), 249–261.

Lopez, P. G., A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere (2015). Edge-centric computing: Vision and challenges. *Computer Communication Review 45*(5), 37–42.

Matt, C. (2018). Fog computing. *Business & information systems engineering 60*(4), 351–355.

Park, J., S. Samarakoon, M. Bennis, and M. Debbah (2019). Wireless network intelligence at the edge. *Proceedings of the IEEE 107*(11), 2204–2239.

Qi, X. and C. Liu (2018). Enabling Deep Learning on IoT Edge: Approaches and Evaluation. *Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing, SEC 2018*, 367–372.

Ren, P., X. Qiao, Y. Huang, L. Liu, S. Dustdar, and J. Chen (2020). Edge-Assisted Distributed DNN Collaborative Computing Approach for Mobile Web Augmented Reality in 5G Networks. *IEEE Network 34*(2), 254–261.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *Preprint, arXiv:1609.04747*.

Shi, W., J. Cao, Q. Zhang, Y. Li, and L. Xu (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal 3*(5), 637–646.

Taleb, T., K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella (2017). On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials 19*(3), 1657–1681.

Tian, Z., C. Luo, J. Qiu, X. Du, and M. Guizani (2020). A Distributed Deep Learning System for Web Attack Detection on Edge Devices. *IEEE Transactions on Industrial Informatics 16*(3), 1963–1971.

Verma, S., Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato (2017). A survey on network methodologies for real-time analytics of massive iot data and open research issues. *IEEE Communications Surveys & Tutorials 19*(3), 1457–1477.

Wang, X., Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen (2020). Convergence of Edge Computing and Deep Learning: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials* (c), 1–36.

Xu, X., D. Li, Z. Dai, S. Li, and X. Chen (2019). A heuristic offloading method for deep learning edge services in 5g networks. *IEEE Access 7*, 67734–67744.

Yang, M. D., H. H. Tseng, Y. C. Hsu, and W. C. Tseng (2020). Real-time Crop Classification Using Edge Computing and Deep Learning. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–4. IEEE.

Zhang, P., Q. Zhao, J. Gao, W. Li, and J. Lu (2019). Urban Street Cleanliness Assessment Using Mobile Edge Computing and Deep Learning. *IEEE Access 7*, 63550–63563.

Zhou, Z., X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE 107*(8), 1738–1762.

Zhou, Z., H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez (2018). Robust mobile crowd sensing: When deep learning meets edge computing. *IEEE Network 32*(4), 54–60.

Mobile clouds: analysis of service models, computation offloading, existing applications

Group 9
Chang Liu
Chenghan Song
Jiacheng Lu

Abstract

Mobile cloud computing is the application of cloud computing technology to the mobile Internet, which refers to the usage pattern of getting the required infrastructure, platform, software, or app in an on-demand, easily scalable way through mobile networks. This essay introduces the evolution of mobile cloud computing over time, such as why new service models need to be referenced and what problems these service models solve. It then embarks on the comparison and analysis of current research schemes of computation offloading. Finally, we conduct a survey of some existing mobile cloud computing applications. Take mobile cloud gaming as a typical case, we analyze and discuss issues and challenges of mobile cloud application.

1 Introduction

Currently, cloud computing and mobile computing are the two research focus of information technology development. Cloud Computing has evolved from distributed computing, parallel Computing and grid Computing. Cloud computing can provide users with the data they need, services they need, and even solutions for hardware facilities via the Internet. Mobile computing is a technology developed with the development of mobile communications, distributed computing, and the Internet. Mobile computing allows data to be transferred through mobile phone or any other wireless device without having to be connected to a fixed physical link. It can deliver information to customers anytime and anywhere, providing users with a ubiquitous mobile computing environment. Those two technologies have dramatically changed the way people live and work.

Mobile cloud computing is based on cloud computing and mobile computing. On the one hand, people will choose more convenient mobile devices such as tablets, smartphones, etc. when using cloud services, rather than being limited to traditional devices such as computers. On the other hand, mobile cloud computing is an extension of the original mobile computing. In the mobile cloud computing model, the huge information processing, complex calculations and massive data storage originally carried out in the smart mobile terminal are transferred to the data center or "cloud", so the hardware and software requirements of the smart mobile terminal is reduced. So the patterns of usage and deployment of various mobile apps have changed dramatically. Although divided into two different areas, smart mobile terminals can combine the benefits of both, together with mobile cloud computing, to create a field with full potential.

"Mobile cloud computing" has come a long way since it was first proposed in 2010. The definition, advantages, and disadvantages of a service model for mobile cloud computing have been discussed in detail in much existing research. For example, the thesis [1] defines the "Device-Channel-Cloud" model and provides a detailed classification of the mobile cloud computing model. And the thesis [2] discusses the service model of IaaS in detail. However, the developmental relationships for each service model layer are not adequately described. Therefore, in the first part of the article, we will

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

explain why each service model layer arises and the problems they reveal in the development of other technologies. This evolutionary sequence will show us the evolution of mobile cloud computing.

As a core technology of mobile cloud computing, computation offloading is the basic solution for mobile terminals to obtain flexible and efficient services. Academia has proposed a series of computation offloading schemes for different optimization goals such as expanding the computing and storage capabilities of mobile terminals, reducing service delay, and saving terminal energy consumption. However, with the aim of designing an efficient and reliable computation offloading service, various measures such as terminal load, task attributes, network status, and application environment need to be considered. Thus, this essay conducted a comparative analysis of the most notable computation offloading schemes. In order to point out differences and challenges of computation offloading, We also evaluated the schemes from the aspects including dynamicity, granularity, optimization goal, performance and disadvantages. It is found out that the bottleneck of offloading is mainly the mobile network technology.

This essay also elaborated on the application examples of mobile cloud computing in detail, on this basis, it deeply analyzes the main problems and solutions in mobile cloud computing application, and finally looked into the application prospects of mobile cloud computing.

The main goal of this essay is to analyze the related research results and development of mobile cloud computing from three aspects: service model, computation offloading and application analysis. Thus, the main contributions are as follows:

- Analyze the reasons for each technical point in mobile cloud computing, explain the problems it solves, and elaborate on the evolution of mobile cloud computing after a sufficient development foundation.
- An analysis and evaluation of existing computation offloading schemes. Unlike previous work, this essay classified the schemes first and analyzed the schemes based on the classification. The comparative Analysis considers key aspects including optimization target, granularity, partition, operating platform.
- This essay surveyed the existing mobile cloud computing applications and take mobile cloud gaming as an example to analyze its characteristics and architecture. Combining with the status quo, this essay identified three major challenges in developing mobile cloud applications and suggested possible solutions.

Our essay is organized as follows. Section 2 analyze the evolution and developing trend of mobile cloud computing. Section 3 performs a survey and compares different computation offloading schemes. Section 4 analyzes and discusses issues and challenges of mobile cloud application. Finally, Section 5 provides the conclusions of the essay.

2 From Virtualization to Mobile Cloud Computing

The cloud computing model includes "device", "channel" and "cloud", as shown in the figure 1 [1]. "Device" means any terminal device capable of accessing the "Cloud" and completing information interactions. "Channel" means the communications network used to complete the transmission of information. "Cloud" means the infrastructure center, platform and application servers, etc,. "Cloud" will be the focus area of this section. This section will describe the evolution of cloud services, starting with virtualization, and will address the reasons why each technology arose and the problems they solve. Secondly, this section will illustrate that with the development of the three parts of the "device-channel-cloud" model, mobile cloud computing has become the focus of academic and business research and has gained astonishing growth in a short period.

2.1 Virtualization

Virtualization is the foundation of cloud computing. Virtualization is defined as a framework or method for dividing the resources of computer hardware into multiple execution environments by applying one or more concepts or techniques (e.g., hardware and software partitioning, time division, partial or complete machine simulation, emulation, quality of service)[3]. On the surface, these virtual machines are independent servers, but in reality, they share the CPU, memory, hardware,

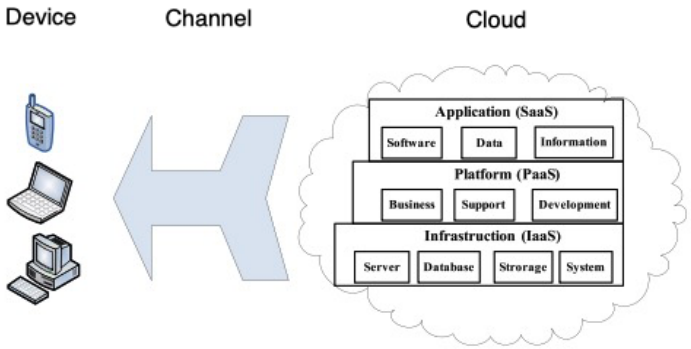


Figure 1: Model of Cloud Computing

network, and other resources of the physical server. The problem of time and space flexibility is solved by virtualization, but virtualization is no longer able to meet demand as clusters grow in size.

2.2 Iaas, Paas and Saas

As clusters grow in size, the process of manually configuring virtual machines becomes increasingly complex and time-consuming. In terms of spatial flexibility, the clusters that can be managed manually are small, and many large companies now have hundreds of thousands servers. It is almost impossible to configure so many machines manually.

To solve this problem, IaaS(Infrastructure as a service) was created. In the IaaS model, the cloud provider hosts the infrastructure components that exist in the data center, including servers, storage, and the virtualization or hypervisor layer. The IaaS provider also provides a range of services to these infrastructure components. These services are policy-driven, enabling users to implement higher levels of automation and scheduling for critical infrastructure tasks. For example, users can enforce policies to drive load balancing to maintain application availability and performance[4].

With IaaS, resource-level flexibility has been achieved, but application-level resilience remains unrealized. Some services require more servers to be configured at a given time. With IaaS, more servers can be created quickly, but the servers are created without any applications and can only be set manually. By the time the server is configured, it's probably past the time of demand. So there's a layer on top of the IaaS for managing application flexibility issues above resources, a layer often referred to as PaaS (Platform As A Service).

With both of these cloud computing, not all needs can be met. For example, if someone doesn't know how to configure a server and just wants to do something with an app, there's nothing that either of these cloud computing service models can do. This is where the SaaS(Software as a Service) layer comes in, where the service provided to the customer is an application running on the cloud computing infrastructure, and the user can access the interface through the client on a variety of devices. Consumers do not need to create or control any cloud computing infrastructure. For example, if a user wants to keep track of the inventory of goods in their supermarket, they can take advantage of the cargo management app provided by the cloud platform. Just simply set up the warehouse information, account number, password and other information, they can use the mobile phone to manage it. There is no need to install the operating system, cargo management software, or complex configuration.

"Device-channel-cloud" model contains three parts, and only when all three components are developed to a certain level can the whole cloud computing have its next breakthrough. With the increasing maturity of SaaS and the continued development of "Device", "channel", mobile cloud computing is ready to be realized.

2.3 mobile computing and mobile cloud computing

Mobile smart terminals (such as smartphones, tablets, etc.) have powerful functions to handle various tasks. They have good internal network protocol backend, which can easily browse the web, access email, instant messaging, and map navigation through wireless network technologies. At the same time, their portability and flexibility are driving the growing demand for cloud services. But pure mobile computing is limited by its hardware and software resources.

- Processing performance limitations: Although CPU performance in smartphones or tablets has improved significantly, the microprocessors used in mobile devices are still inadequate for complex tasks[5].
- Capacity constraints: Most of the storage used in mobile devices is flash memory, which, in today's data explosion, it is still not large enough to support users' vast storage needs.
- Power limitations: batteries have been a key constraint to the development of miniaturization of smart devices, and ensuring the longevity of mobile devices is always a challenge.
- Data reliability: In an unstable mobile network environment, there is still no reliable solution to ensure real-time data reliability within mobile terminals.

To address these issues, a new cloud service model - Mobile Cloud Computing (MCC) - has emerged. The following characteristics of MCC can solve the above shortcomings.

- Move tasks that consume relatively large amounts of computing resources to the cloud for processing, and transfer data with large amounts of information to a cloud storage center. Small tasks are handled directly in the mobile client. In contrast, the large tasks only retain a simple interface, and the mobile client is only responsible for data entry and query of results.
- The cloud has much more data storage capacity compared to a single mobile device. The emergence of mobile cloud computing can meet the need for mobile users to directly store or access large amounts of data through a wireless network in their mobile device. For example, iCloud and OneDrive, can save users a lot of storage space.
- In a mobile cloud environment, mobile devices migrate tasks to the cloud, reducing the load on the device itself, thereby reducing energy consumption and extending battery life.
- The data is stored in the cloud, and there are usually multiple computers backing up the data to ensure reliability. Also, a well-established system will often have strong security protection measures to safeguard the privacy of user data.

While mobile cloud computing has many advantages, it still faces some problems when applied in practice.

- Stability of wireless connectivity: When users use cloud-provided applications, there is a possibility of signal outages due to geographic location, or they may not be able to connect to the cloud for cloud-provided services due to congestion. Therefore, to ensure the mobile application can continue to provide a stable service experience for users, it is also necessary to ensure the wireless connection's stability.
- Mobile cloud computing security is more complicated than traditional cloud computing security, mainly because mobile terminals have flexible access locations, a higher number of concurrent users and mobile devices are easier to lose and leak information. To address these challenges, developing a mobile cloud security strategy and mobile cloud security services became the new direction of study.

The problem of stability of wireless connectivity can be solved by popularizing 5G technology and building more mobile base stations. The security issues in mobile cloud computing will be solved with the general public care about privacy and more advanced encryption methods.

3 Computation Offloading

As the core technology of MCC, computation offloading[6][7] mainly solves the problem of limited computing and storage resources of mobile terminals. In the form of entire applications[8][9] or

part of code/data[10][11], it offloads mobile terminal storage and computing tasks to cloud data centers. Execution on high-performance servers is a basic method for mobile terminals to obtain flexible and efficient services. Academia has proposed a series of computation offloading solutions for different optimization goals, such as expanding the computing and storage capabilities of mobile terminals, reducing service delay, and saving terminal energy consumption. However, in order to design an efficient and reliable computation offloading system, various measures such as terminal load, task attributes, network status, and application environment need to be considered. In recent years, academia has also paid attention to relevant research.

Computation offloading process mainly contains 5 parts: surrogate discovery, environment awareness, task division, task scheduling, and execution control[6]. However, not every computation offloading scheme includes all steps. The most important execution control mainly involves how to connect to a reliable remote surrogate, pass the required information for execution, execute remotely and return the result.

Computation offloading schemes are generally classified according to the granularity of division, mainly including fine-grained computation offloading based on processes and function functions, and coarse-grained computation offloading based on applications and VMs. Computational offloading schemes are generally classified according to granularity, mainly including fine-grained computation offloading based on processes and functions, and coarse-grained computation offloading based on applications and VMs.

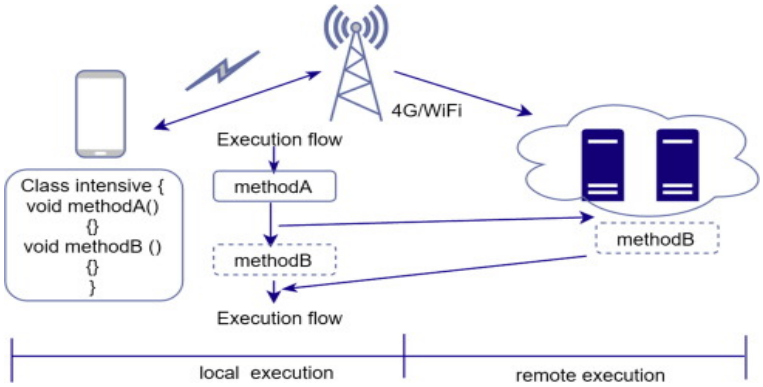


Figure 2: Offloading process overview

3.1 Fine-grained offloading

The fine-grained computation offloading scheme offloads some computationally intensive code or functions in the application to the cloud to execute. Such schemes require developers to divide the program in advance by marking and modifying the code. According to different offloading strategy, fine-grained computing offloading can be classified into two types: static partition and dynamic partition. The static partition scheme offload according to the programmer's pre-annotation rule, while the dynamic partition scheme can dynamically adjust the partition of offload areas according to real-time changes of system load, network bandwidth, and other factors. Dynamic partition can relatively improve efficiency and reliability.

3.1.1 Static partition

Most early computation offloading technologies used static partitioning schemes. Programmers have to divide the application into two parts previously. While running the program, the first part will be executed on the mobile terminal and the other part will be executed on the remote server. In Protium[12], the programmer divides the application into a viewer part and a service part. The viewer part runs on the mobile terminal, and the service part runs on the proxy server with abundant storage capacity and CPU computing resources. The application can define protocol of communication between two parts. If the program contains complex cross-state and viewer management, then the program needs to be rewritten, which increases the burden on programmers. Moreover, it is

impossible for the programmer to grasp the dynamically changing network status and the energy consumption of the program on CPU and memory accurately. Therefore, this static labeling partition scheme can not guarantee that the energy consumption of program execution is minimized.

The solution proposed by Yang et al.[13] takes the utilization of multiple resources, including CPU, memory, and communication costs (such as bandwidth resources) into consideration, and seamlessly offloads some tasks on mobile terminals to surrogates. This user-server offloading structure contains modules such as monitors and offloading engines. The resource monitor is responsible for monitoring the usage of various resources such as memory, CPU and current wireless bandwidth. The offloading engine divides the application into a local part and multiple remote parts. The class method module converts the class into a method module that can be executed remotely. The scheme divides the application into (k + 1) divisions, including 1 non-migratable division and k disjoint and migratable divisions. Then these divisions will be organized into a directed graph, the vertex set represents the Java class, and the edge set represents interaction between classes. Their algorithm can give a offloading scheme close to the optimal solution based on the graph.

In order to further improve the execution efficiency of computation offloading , Misco[14] implements a clustered service and supports the distribution of data to multiple nodes on the network to process application data in parallel. The master server play the role as a centralized monitor, which is responsible for the implementation of MapReduce. The application program is statically divided into two parts: Map and Reduce. The mapping function processes the input data, generates intermediate key-value pairs, and classifies all generated key-value pairs to form corresponding data block nodes. All data block nodes produce the final result through the reduction function and return it to the main server. While developing the application, developers have to confirm map and reduce functions to provide a distributed platform. However, the system simplified the problem of data locality, decentralization, and device heterogeneity.

Most of the static partition schemes assume that the communication cost and calculation time can be obtained by prediction or statistics before processing. Once the partition scheme is determined, it will remain unchanged. However, due to the differences between mobile terminals and the complexity of network status, it is predict the cost of calculation and communication accurately.

3.1.2 Dynamic partition

In order to overcome the shortcomings of the static partition, the dynamic partition scheme can dynamically adjust the partition of the offloading area according to the real-time status of network bandwidth, system load, etc. Dynamic partition comprehensively considered the available resources to improve the efficiency and reliability.

The solution proposed by Chun et al.[15]. comprehensively considers the changes of three factors: mobile terminal power, network connection status, and real-time bandwidth. They provide different solutions according to different changes of these three factors and design a universal formal model for the offloading decision problem. However, they do not give any detailed system design or implementation.

Afterwards, academia proposed a series of computation offloading systems for specific applications. For example, CogniServe[16] for image recognition and speech recognition applications, Odessa[17] for environment-aware applications, Sociable Senses[18] for social applications, and Kahawai[19] for cloud games.

The aim of MAUI[10] is to provide a general dynamic offloading service which can minimize the burden on developers. Developers do not have to make offloading decision for each program, instead, they only need to classify the application into local methods and remote methods. Based on the collected network status and other information, the MAUI event analyzer dynamically decides which remote methods need to be offloaded to the cloud for execution dynamically. Then the surrogate execution module executes the corresponding control and data transmission work. This system evaluates the network status by sending 10KB of data to the server. However, its prediction still needs to be improved if the wireless network is unstable and changes dramatically.

ThinkAir[11] is also a thread-level dynamic offloading scheme which focuses on enhancing the server. This system can dynamically allocate resources such as service memory for offloading tasks, which improves the reliability of system operation. Based on MAUI and ThinkAir, Comet[20] uses distributed shared memory and VM-synchronization primitive to support multi-threaded parallel

offloading, which enables free migration between machines depending on the workload. The computing offloading system designed by Zhou et al.[21] can make dynamic decisions based on the context of wireless channels, cloud resources and other contexts when the program is running, and dynamically select servers in multiple clouds such as micro-clouds and public clouds to achieve code-level refinement. Granularity calculation offloading. The problem of this offloading decision algorithm is the intercommunication between different cloud resources.The prototype's performance can be further optimized by considering more context parameters.

In order to overcome the mismatch between the demand and offer of computing resources. Shi C et al.[22] designs and implements the Cosmos system. Cosmos provides computation offloading as a service to mobile devices. It can allocates and schedules the offloading requests and makes offloading decisions in a risk-controlled manner. To some extent effectively, Cosmos solves the problems of uncertainty caused by variable network connectivity and program execution.

From another aspect, fine-grained offloading leads to the consumption of additional partitioning decisions, so the quality of the partitioning algorithm directly affects the offloading efficiency, and the optimal solution is not always obtained. In addition, both programmer modify code and remote execution management will introduce additional overhead. Thus, it is possible to result in more consumption of CPU energy.

3.2 Coarse-grained offloading

Coarse-grained offloading encapsulates the entire application in a VM and sends it to the cloud server for execution. In this way,the extra cost of program partition and offloading decision-making can be reduced.

Cyber Foraging[23] uses nearby computers with strong computing capability as proxy servers to provide computation offloading services for mobile terminals. Once the application starts running, the mobile terminal will send a offload request to the service search server. The server returns available surrogate's IP and port number. The mobile terminal can then apply to the corresponding agency service for computing offloading services. Each surrogate runs multiple independent virtual services to make sure that each application's virtual service space is isolated. Cyber Foraging utilizes local area network to provide mobile terminals with efficient computing and offloading services. However, deployment methods based on surrogate discovery and VM templates cost much time and resource.

Clonecloud[9] also uses VM to establish the operating environment directly in the cloud, without any additional changes to the operating system and applications. It designed three different offloading algorithms for different types of applications. In addition to offloading computation-intensive tasks such as voice recognition and image processing to the cloud, security detection is also migrated to the cloud server, which reduces the burden on the terminal. However, the strategy based on application diversity increases the overhead of mobile terminals. The single-thread deployment method also makes the system unstable.

In order to overcome the disadvantages such as unstable wireless network, Tango[24] deploys multiple copies to perform calculation tasks simultaneously on the server and mobile terminal, and uses the fastest execution result as output, which further improves the reliability of the system.

Table 1 shows the overall comparison and analysis of the above computation offloading schemes.

Table 1: Comparison of different computation offloading schemes

Name	Operating platform	Target	Granularity	Partition
Misco	Mobile cloud node	Delay	Method Level	Static
MAUI	Cloud Server	Energy Consumption	Method Level	Dynamic
ThinkAir	Cloud Server	Delay/Energy consumption	Threadlevel	Dynamic
Comet	Cloud Server	Delay	Multi-thread	Dynamic
CogniServe	Locally distributed	Performance/Energy consumption	Application	-
CyberForaging	Local-distributed	Delay	Application	-
Clonecloud	Cloud Server	Performance/Energy Consumption	VM	Dynamic

4 Application Analysis

4.1 Mobile cloud computing applications

In recent years, with the diversification of mobile terminals, especially the prevalence of smartphones, and the continuous development of mobile cloud computing, many mobile applications have taken advantage of mobile cloud computing and occupied an increasing share in the mobile market. Mobile cloud computing allows mobile devices to offload applications to the cloud which improves the quality of service and meets people's high demand for availability and mobility. Users can access the application via the wireless networks with no need for large storage capacity of mobile devices which overcome the resource limitation[25]. Typical examples of mobile cloud computing applications include mobile cloud storage and mobile commerce, and mobile cloud computing technologies are also incorporated into services such as multimedia sharing and mobile learning. Besides, mobile cloud computing has also been applied in the fields of cloud-based mobile healthcare applications, mobile gaming, and mobile social networking[26].

- Mobile commerce

  Mobile commerce utilize scalable computation and storage in the cloud. Mobile banking, mobile advertising, mobile ticketing, and e-shopping are typical examples of mobile commerce[26]. During the process of mobile commerce, security measures are of vital importance. M-commerce may be interfered by external factors that lead to the leakage of personal private information and the user's economic interests can be seriously threatened. This is a relatively serious problem that needs to be resolved urgently.

- Mobile learning

  In traditional mobile learning, instability of transmission rate and high cost of devices limited the development of e-learning. The implement of mobile cloud has solved these shortcomings and users are able to easily access learning materials on the cloud via mobile devices[25]. With the advent of the 5G era, mobile communications can provide faster speeds as well as more reliable and stable connections. In mobile learning, 5G connections will further accelerate the connection speed, and slow buffering will become a thing of the past. We can look forward to the rapid growth of more high-quality audio and video content streams in mobile learning.

- Mobile Healthcare

  The global healthcare industry is undergoing a transformation. Industry convergence, increased customer expectations, aging population, and complex safety and regulatory requirements have brought a lot of pressure to the healthcare industry. Mobile cloud computing plays a very important role in constantly driving industry progress. Mobile healthcare can store a great amount of data on the cloud instantly. It allows doctors to have access to the records of patients on mobile phone. With a healthcare monitoring system which collects patients' physiological parameters from wireless sensors, doctors can diagnose and monitor the real-time health status of patients remotely[27].

4.2 Mobile cloud Gaming

The video game industry has changed dramatically in recent years. On the one hand, traditional computer games have a better performance compared to mobile games. However, the rapid updating of hardware equipment and high game content fees have greatly increased user costs and affected the development of the game industry. On the other hand, mobile games have gained a considerable market in recent years due to their excellent portability. With the increase in the number of mobile users, many companies have launched their mobile cloud gaming platforms, hoping that mobile cloud games will combine the high quality of computer games and the portability of mobile games.

Mobile cloud gaming migrates the complex computing of traditional games to the cloud. The cloud performs tasks of game calculation and data storage, encoding the game into real-time video streaming to the mobile terminal. This not only greatly expands the execution capabilities of mobile terminals, but also improves the platform compatibility and the flexibility of upgrade and maintenance of games. In this section, we will analyze mobile cloud gaming.

4.2.1 model of mobile cloud gaming

An overview of design patterns used to host game infrastructure is illustrated in Figure 3, the cloud server initiates virtual machines (VMs) to simulate the runtime environment of the mobile devices[28]. The virtual machines are composed of game engine servers and game streaming servers. The game content server will first confirm the connection between the user and the game server. Then it will initialize the game engine server, the game engine server will load account information of the user and game data from the content server, and starts to process the game logic and user data to render the game[28]. The inputs of users are sent to the cloud server and accepted by the content server directly.

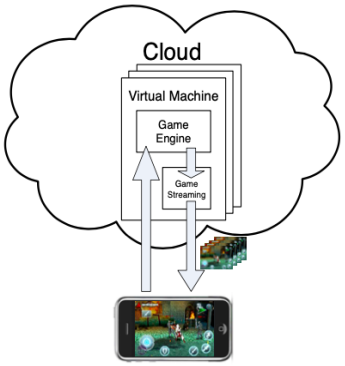


Figure 3: Mobile Cloud Gaming

4.2.2 Open challenges

Partition

Mobile cloud gaming systems ought to be divided into components in order to achieve onloading and offloading. As gaming is process-oriented, dynamic partitioning can be implemented in mobile cloud gaming[28]. In addition, the components should be adapted to different client systems. How to partition efficiently is still under research.

Latency and Connectivity

1) Due to the huge amount of information processed in mobile cloud computing, there has been a certain degree of delay in the calculation process. Mobile cloud computing responds to network delays quite sensitively. To provide players with a ubiquitous gaming experience, [28] proposed a solution to implement delay-tolerated mechanisms in game design. Take Draw Something as an example, users access to the drawings of teammates through network. Once the network is disconnected, the status will be frozen.

2) With the advent of the 5G era, the problem of latency will be effectively solved. On the one hand, 5G carries high bandwidth. In the 4G era, Video live broadcasts can be realized. With the implementation of 5G, applications such as cloud gaming and VR/AR live broadcasting will breakthrough network bandwidth limitations and will be used on a large scale. On the other hand, 5G achieves low latency. For mobile cloud gaming, if the delay exceeds 100 milliseconds, the user's operating lag will be very strong, which will greatly affect the user's gaming experience. 5G will allow the delay to be within 10 milliseconds[29], building the best low-latency environment. But currently, the global 5G network has just started commercial use, and will take some time before it is popularized. According to the report from the GSMA Intelligence[30], there are nearly 40 mobile network operators in the world which have officially commercialized 5G, and the number of global 5G users is only about 5 million.

Cost

1) For cloud application providers, cloud service costs mainly include server, IDC, and bandwidth. Among them, server costs account for a relatively high proportion. The future cost control focuses on the software and hardware costs of cloud servers. With the continuous development of cloud

gaming technology, the efficiency of hardware utilization will be greatly improved, and the server will provide more instance concurrent thread services, which will reduce server costs exponentially.

2) The main goal of mobile cloud computing is to save the energy of mobile devices. However, in some cases, it might cost more energy consumption to offloading a simple task to the cloud than run it in the mobile device. Thus the concept of Green mobile cloud computing is proposed to improve energy efficiency. Lu et al.[31] suggested to compress data by excluding redundant data to reduce the volume of transmitted data. [32] proposed different techniques for green applications. By converging green cellular networks with the cloud environment, green mobile cloud computing is achievable.

5 Conclusions and future work

With the wide application of wireless data communication and mobile Internet, mobile cloud computing technology[33] has developed rapidly and has attracted widespread attention from scholars.

This paper introduced the whole evolution process of mobile cloud computing, explains why each service model layer arises and the problems they reveal in the development of other technologies. The goal is to help understanding the development trend of mobile cloud computing. Mobile cloud security will be the next focused areas.

In order to help design an efficient and reliable computation offloading service, we also researched and analyzed computation offloading, which is the core of mobile cloud computing. The paper classified the computation offloading schemes first and then comparatively analyzed most mainstream schemes from aspects such as optimization target,granularity, partition. Through analysis, we concluded the disadvantages for different kinds of offloading scheme. For fine-grained computation offloading, programs have to be pre-divided and labeled, and only parts that can save resources by remote execution will be offloaded. For coarse-grained computation offloading, its applicability has certain limitations[10]. For example, this kind of scheme cannot apply to programs that has frequent interactions with users.

Moreover, we presented a survey of existing MCC applications and take mobile cloud gaming as an example to analyze its characteristics and architecture. In addition, this essay identified three major challenges in developing mobile cloud applications and suggested possible solutions, hopefully can provide insights for the future research.

As future work, we intend to survey existing concepts integrating Mobile edge computing functionalities to the mobile networks and make a comparison between MEC and MCC using our methodology.

References

[1] Bo Lang Hanbing Yan Li Ding Jia Li Yu Zhou Xiaochun Yun Weiping Wen. “Cyber Security: 15th International Annual Conference”. In: 2018, pp. 167–168.

[2] Naibo Yu. “Cloud computing IaaS service model explored”. In: 2011.

[3] David Rule Rogier Dittner. “The Best Damn Server Virtualization Book Period”. In: 2007, Chapter: An Introduction to Virtualization.

[4] Margaret Rouse Michelle Boisvert Stephen J. “Infrastructure as a Service (IaaS)”. In: URL: `https://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS`.

[5] Lee D Huerta-Canepa G. “A virtual cloud computing provider for mobile devices”. In: 2010, Chapter: Proceedings of Acm Workshop on Mobile Cloud Computing Services.

[6] Mohsen Sharifi, Somayeh Kafaie, and Omid Kashefi. “A Survey and Taxonomy of Cyber Foraging of Mobile Devices”. In: *IEEE Communications Surveys Tutorials* 14.4 (2012), p.1232–1243.

[7] W. L. Zhang et al. “Computation offloading on intelligent mobile terminal”. In: *Chinese Journal of Computers* (2016).

[8] Satyanarayanan et al. “The Case for VM-based Cloudlets in Mobile Computing”. In: *IEEE Pervasive Comput* (2011).

[9] Byung Gon Chun and Petros Maniatis. “Augmented Smartphone Applications Through Clone Cloud Execution”. In: *Proceedings of HotOS’09: 12th Workshop on Hot Topics in Operating Systems, May 18-20, 2009, Monte Verità, Switzerland*. 2009.

[10] Eduardo Cuervo et al. “MAUI: Making smartphones last longer with code offload”. In: *International Conference on Mobile Systems*. 2010.

[11] Sokol Kosta et al. “ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading”. In: *Proceedings IEEE Infocom* 945-953 (2012), pp. 945–953.

[12] C. Young et al. “Protium, an infrastructure for partitioned applications”. In: *Proceedings Eighth Workshop on Hot Topics in Operating Systems*. 2001, pp. 47–52.

[13] Kuanli Yang, Shumao Ou, and Hsiao-Hwa Chen. “On effective offloading services for resource-constrained mobile devices running heavier mobile Internet applications”. In: *Communications Magazine, IEEE* 46 (Feb. 2008), pp. 56 –63. DOI: 10.1109/MCOM.2008.4427231.

[14] Adam Dou et al. “Misco: A MapReduce Framework for Mobile Systems”. In: Dec. 2010. DOI: 10.1145/1839294.1839332.

[15] Byung Gon Chun and Petros Maniatis. *Dynamically Partitioning Applications between Weak Devices and Clouds*. ACM, 2010.

[16] Ravi Iyer et al. “CogniServe: Heterogeneous Server Architecture for Large-Scale Recognition”. In: *Micro, IEEE* 31 (July 2011), pp. 20 –31. DOI: 10.1109/MM.2011.37.

[17] Moo Ryong Ra et al. “Odessa: Enabling interactive perception applications on mobile devices”. In: *International Conference on Mobile Systems*. 2011.

[18] Kiran K. Rachuri et al. “SociableSense:exploring the trade-offs of adaptive sampling and computation offloading for social sensing”. In: *International Conference on Mobile Computing Networking*. 2011.

[19] Eduardo Cuervo et al. “Kahawai: High-Quality Mobile Gaming Using GPU Offload”. In: *International Conference on Mobile Systems*. 2014.

[20] Mark S. Gordon et al. “COMET: Code Offload by Migrating Execution Transparently”. In: *Usenix Conference on Operating Systems Design Implementation*. 2012.

[21] Bowen Zhou et al. “A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service”. In: *IEEE CLOUD 2015*. 2015.

[22] Cong Shi et al. “COSMOS: computation offloading as a service for mobile devices”. In: Aug. 2014. ISBN: 978-1-4503-2620-9. DOI: 10.1145/2632951.2632958.

[23] S. Goyal and John Carter. “A lightweight secure cyber foraging infrastructure for resource-constrained devices”. In: Jan. 2005, pp. 186–195. ISBN: 0-7695-2258-0. DOI: 10.1109/MCSA.2004.2.

[24] Mark S. Gordon et al. “Tango: Accelerating Mobile Applications through Flip-Flop Replication”. In: *Getmobile Mobile Computing Communications* (2015).

[25] Debashis De. “MOBILE CLOUD COMPUTING: Architectures, Algorithms and Applications”. In: 2016, pp. 71–74,303–320.

[26] Wang D C Wang Y Chen R. “A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges”. In: *Wireless Personal Communications* 80.4 (2015), pp. 1607–1623.

[27] Loai Tawalbeh et al. “Mobile Cloud Computing Model and Big Data Analysis for Healthcare Applications”. In: *IEEE Access* PP (Sept. 2016), pp. 1–1. DOI: 10.1109/ACCESS.2016.2613278.

[28] W. Cai, V. C. M. Leung, and M. Chen. “Next Generation Mobile Cloud Gaming”. In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. 2013, pp. 551–560.

[29] China Information, Communication Research Institute, and 5G Cloud Game Industry Alliance. “Cloud Game Industry Development-5G Helps the Rapid Development of the Cloud Game Industry”. In: 2019.

[30] GSMA. “2020 The Mobile Economy”. In: URL: https://www.gsma.com/mobileeconomy/#key_stats.

[31] Y. Wang X. Lu E. Erkip and D. Goodman. “Energy efficient multimedia communication over wireless channels”. In: *IEEE Journal on Selected Areas in Communications*. Vol. 21. 10. 2003, 1738–1751.

[32] M. Chen Y. Liu X. Wang A. V. Vasilakos and T. T. Kwon. “A survey of green mobile networks: Opportunities and challenges”. In: *Mobile Networks and Applications*. Vol. 17. 1. 2012, pp. 4–20.

[33] LUO et al. “Mobile Internet: Terminal Devices, Networks and Services”. In: *Chinese Journal of Computers* (2011).

Table 2: Division of tasks.

Task	Name
Analyze the evolution of mobile cloud computing	Chang Liu
Research of service model of mobile cloud	Chang Liu
Categorize the current research work of mobile cloud computing	Jiacheng Lu
Research of Computation Offloading	Jiacheng Lu
Application Analysis of mobile cloud computing	Chenghan Song
Research of current problems and possible solutions of MCC applications	Chenghan Song

Cloud-Based Payment Systems: A Literature Review

Alex Boyko
Leyu Liu
Yizhen Zhao
University of Amsterdam

June 1, 2020

Abstract

With the evolution of market needs and the increase of business migrations to online environments, there has been a constant demand for new payment solutions that produce flexible systems that suit customer needs. Over the years the adoption of the cloud services has been progressing rather slowly, yet as more and more businesses migrate from legacy systems to cloud environments, it is apparent that the future of the payment software relies on cloud services. This paper evaluates the state of cloud-based payment systems by reviewing and analyzing the leading concepts of payment services and Cloud Computing. A comprehensive analysis of the cloud-based payment systems literature consists of a number of research topics with common ideas shared among papers. A combination of research topic contributes to forming a broad perspective of cloud-based payment services. The paper concludes with the answers to three research questions regarding present and future payment systems that employ cloud services.

1 Introduction

Throughout history, there are lots of changes in how people would like to pay. The payment system also goes through migration from offline to online. For a long time, people preferred to use cash to pay for everything in life. However, recently, using debit or credit cards has become one of the most popular payment methods. It shows that around 78% of purchase is done through debit or credit card [1]. With the development of the digital world, people became more interested in Cloud Computing. By using the Internet's ability to store more information in the cloud, service providers could deliver efficient payment services to customers.

The cloud started a few years ago, just like any other new industries, it would take some time for people to accept it. One of the researches mentioned that the future of payments is the cloud [2]. The banking clients started their cloud

journey by using platforms like Salesforce and Microsoft Dynamics in sales and marketing. They observed success in using clouds in their business. Moreover, the rapid growth of cloud-based payments providers like PayPal, Stripe and etc., gain higher values than many other major banks. That is one reason why many banks are trying to use the cloud to support their customer services. Payments would be on top in the list for "cloudification".

Why clouds become popular in people's lives? Why using the payment system in clouds is the future? There are lots of reasons behind [3]. One of the key advantages is flexibility. Cloud-based payment system could make people's lives easier. It allows payments to be made through mobile phones, computers, bank transfer through the Internet, etc. All of the operations could be paperless and transaction records could be saved and sent directly to customers' emails. Another advantage is that it improves the scalability. Cloud-based payment stores payment data via the data center, therefore, we do not need to worry about the capacity issue. This also benefits small or medium-sized businesses, as they only pay for what they use. With the help of the Internet and cloud, such smaller businesses could achieve the same efficiency as big businesses, but with affordable prices. Besides, a cloud-based payment system is also more secure and easier for data integration. In summary, the key idea of cloud-based payment system is to enhance customers' experience in payment.

The main objective of this paper is to conduct a literature review of cloud-based payment systems, where we intend to find out the current state of cloud-based payment, the characteristics and main trend of this area, and the possible future research directions. Thus, we identify the core concepts by literature classification and analyze existing methods to form a deep understanding of cloud-based payment. Additionally, we make comparisons from various perspectives and point out the results, research gaps and further directions.

We organize this paper into four sections. First, we introduce background information here in the ***Introduction***. In the following, we describe the research methodology, define the research questions and literature groups, and evaluate the selected literature in ***Methodology***. Then, in ***Discussion of Literature***, we classify the literature and conduct a detailed analysis for each literature group to answer the research questions. Lastly, the discussion about the future research and the summary of this paper are included in ***Conclusion***.

2 Methodology

A literature review normally summarizes, analyzes, synthesizes and evaluates the existing literature in a certain research area and provides suggestions for the future research[4]. The research methodology of this paper follows Webster et al.[5], Stieninger et al.[6], Novais et al.[7] and Briner et al.[8]. Therefore, this review has been conducted through the following steps: (1) Define the research topic and research questions; (2) Select relevant papers and online articles; (3) Analyze and synthesize the contents of all literature; (4) Classify literature into groups of topics; (5) Discussion of the literature; (6) Conclude the review and

discuss further research opportunities.

The topic we chose for this literature review is "payment systems in clouds (models used by Cloud Service Providers) and for cloud-based application", which can be interpreted as the payment systems that use the cloud environment to operate, in another word, cloud-based payment systems. Therefore, in the first step, we define our research questions as follows:

- Q1: What is the current state of cloud-based payment?
- Q2: What are the characteristics and current research trends of cloud-based payment?
- Q3: What are the possible future research directions for cloud-based payment?

In the second step, to answer the research questions, we selected relevant literature for 9 scientific publications and 6 credible online articles. For published papers, we used searching tools, such as Scopus and Google Scholar, to select peer-reviewed papers that have a relatively large number of citations. The papers were also selected based on whether they were published on well recognized conferences and journals including IEEE International Conference on Cloud Computing and Big Data Analysis, Proceedings of the 3rd International Conference on Business and Information Management, International Conference on Cloud Networking, Computer Standards & Interfaces, Computer Law & Security Review, and Journal of Systems and Software. Apart from it, the dates of the publication are ranging from year 2015 to year 2019, where we tried to make the findings as recent as possible to match with the research questions. For online articles, they were first identified by searching with the keyword "cloud-based payment" and then we selected those that were highly relevant to cloud-based payment with a specific focus, such as its importance, popularity, trend and definitions.

In the third step, we analyzed the contents and synthesized the literature according to the concepts, characteristics and research directions. Then, based on the results of the analysis and synthesis, we defined the classification groups for the selected literature, which include ***Definitions***, ***Popularity and success factors***, ***Models***, ***Banking in the clouds***, ***Security and privacy issues*** and ***Innovative features***. The discussions of relevant findings of the literature is presented under each specific topic. As for the last step, a conclusion of this paper is identified where the implications and future research are being addressed as well.

3 Discussion of Literature

We intend to answer the research questions defined above throughout this literature review. The selected literature was grouped into 6 topics, where the analysis of each paper or article under different topics reflects the current state and characteristics of the cloud-based payment systems(Q1, Q2) and reveals the

research gaps, trends and future research directions(Q3) that will be summarised in conclusion. For definitions, 1 paper and 1 online article were reviewed. Besides, we analyzed 2 papers and 3 articles that were related to the popularity and success factors behind cloud-based payment. There are 6 papers which were selected to describe the models used for cloud-based payment system under different scenarios. In addition, we found 1 paper and 2 articles under the topic of banking in the clouds and 5 papers and 1 article focused on the possible security and privacy issues. Lastly, 2 articles were reviewed that illustrated the innovative features in cloud-based payment systems.

3.1 Definitions

Clouds have its applications everywhere nowadays. Virtualization is finding ways to emerge into people's daily operations. Cloud computing provides a large capacity of data storage and online access to internet services, as it groups remote servers and computers. A survey [9] showed that cloud-based payment system integrates the ability of cloud computing and business' existing accounting system. Using a cloud-based payment system allows businesses to deal with checks, credit cards, cash, etc. It brings better user experience, as it provides easy to use interface and multi-channel access such as computer and mobile device. Cloud-based payment systems improve flexibility and scalability. It also improves security, as it provides different types of authentication methods.

With the rapid development of mobile technology, it also triggers the appearance of mobile commerce (m-commerce). According to Marwah's [10] research, m-payment's aim is to integrate mobile devices with different kinds of billing system or payment system. Cloud-based mobile payment utilizes cloud computing's ability such as resource sharing and content sharing. In m-payments, NFC is wildly used as it leverages the Radio Frequency IDentification (RFID) technology to enable two devices to exchange information within a short distance. Mobile cloud moves heavy computing tasks into the cloud, deploys services and applications on the cloud so that cloud-based payments can have a high performance and reliability.

3.2 Popularity and success factors

As we mentioned before, cloud-based payment methods are getting popular in our life. Nowadays, people tend to order food via Uber Eats app and use Apple Pay to pay for their food. Big companies, such as Google, Facebook and Amazon started benefiting from the convenience of cloud-based payment systems.

Sulabh Agarwal[2] highly recommends cloud-based payments especially for banking business, given the reasons that it has the benefits of resilience, scalability, flexibility and agility. In addition to that, Jessica Travis[1] mentions that unlike virtual terminal, which is a web-based point-of-sale system for virtual credit card payments, the payment gateway technologies have higher volume of transactions and can be integrated to any software. It also allows to avoid manual operations

when running a big business remotely, which makes it a better tool under many circumstances with such convenience and efficiency.

Qin et al. [11] agree that mobile payment has become one of the most frequently used approaches to provide payment services to the modern society. In turn mobile payments are highly dependable on the cloud environments, which offer them the ability of taking over the heavy computation workload from the resource constrained mobile devices. That is not surprising as the sole purpose of clouds is to provide a robust environment that is capable of delivering better performance and scalability for large digital systems. The addition of cloud services has allowed mobile payments to scale well with the current business needs, yet it has imposed new security challenges that will be discussed further as a part of Security and privacy issues.

Ni et al. [12] mention that Cloud-based payment system is changing the payment market. To some extent, it brings huge convenience to human society. Mobile payment is a new and modern method of payment that uses a mobile device. It can be used for goods and services, also for lots of other technologies, such as internet banking, direct debit transfer, etc. A survey [13] showed that nearly 64% of consumers have online purchase records in the past 12 months. Among those digital transactions, nearly 40% is provided by Apple Pay, followed by PayPal, which takes up 35% of total digital transactions. However, there are lots of factors that influence people's acceptance of cloud-based payment, where the quality of the service might be an important one. Whether this service provides a good user experience or whether this service is safe to use. The social influence could be another factor, as the reason why so many people are using Apple Pay might be because of its popularity and the reputation Apple has on the market.

3.3 Models

Different types of models are applied in cloud-based payment. Each of them has its own features and characteristics, and suits for different scenarios.

Ni Zeng[12] introduces four main cloud-based payment models in his research, including Premium SMS based transactional payments, Direct Mobile Billing, Mobile Web Payments, and Contactless NFC. The most promising model would be NFC technology. Payments using NFC enable two electronic devices to exchange information and complete transactions when they are close to each other, normally within 4cm. It is estimated that NFC-enabled smart device shipments will reach 2.2 billion by 2020. For instance, Apple Pay received a huge success in the public since 2016, which is to leverage the advantages of cloud-based payment methods.

A payment model proposed by Jem-Ho et al. [14] was applied to handle security and privacy issues. The structure is described as follows. First, consumers' privacy has been protected, since a temporary identity is used to communicate with the merchant and information about products has been packed using hash function. Secondly, the client's payment bank would generate a digital signature as a proof in case some malicious clients do not pay for the products. Finally,

it decreased the computation and communication costs by computing payment relevant information (i.e. time, date, the amount of money) and packs as a message, just uses public key to validate the relevant information.

Madhoun et al[15] propose a model to deal with security vulnerabilities for cloud-based mobile NFC payment, which includes 7 steps for message exchanging between cloud infrastructure, NFC smartphone and NFC payment terminal. The first step was for the payment terminal sending to the smartphone authentication request messages, which used the terminal certificate, acquiring bank's certificate and hashed secret key of the terminal to ensure the integrity of the message. Then the smartphone will send an encrypted text message of the terminal authentication and session requests to cloud infrastructure, which would decode the message, try to authenticate payment terminal when the timestamp was valid and send the authenticity and session confirmations message back to the smartphone. After the smartphone passed the authentication message to the terminal, a confirmation message would be sent from the terminal to the smartphone and then from the smartphone to the cloud infrastructure. After the payment transaction was confirmed, the payment session would start securely between the smartphone and payment terminal with the electronic signature to ensure integrity.

Liao et al[16] introduced a security model to improve Qin et al's security protocol[11], where Liao et al.[16] claimed to solve the colluding attack during the outsourced verification phase. The model includes three phases. First, a master key and a public key are generated by payment service provider during the setup, after which a pseudo identity of the customer and a short-time partial key would be generated. Then during the payment transaction, customer A could generate a signature upon receiving the payment request from customer B with the private key and then A would receive and verify the signature-receipt pair from B. Lastly, customer B would verify the messages sent by the cloud server verification provider(CSVP) based on the random elements generated by B during the outsourced verification phase to complete the transaction. Kang et al.[17] pointed out the potential flaws of Liao et al's model that a colluding attack could still happen if the customer used fake payment information and identity and [16] proposed a new model to prevent such attacks that modified the outsourced verification phase using random numbers and hash function regarding the information sending to CSVP.

3.4 Banking in the clouds

As a cloud-based payment system becomes more and more popular, it also finds a place in the banking industry. Flexibility is a key reason why the cloud-based payment system is popular in banking. Scalability is also recognized as a major benefit for the banking system. But there is usually a long way before legacy system can migrate its services to the cloud.

Marc Sczesnak in his article [18] talks about how the migration of payment systems to the cloud is becoming more popular due to a number of benefits that come with using a private cloud environment over in-house data centers. He

identifies the key attributes that make cloud environments more appealing for the banks, such as lowered costs, improved speed to market and catching up with customer demand for new products and services. Being able to adapt and provide necessary flexibility is something that is being highly valued not only among payment services, but for any modern service.

A study done by Kuan Hon et al. [19] gives an in-depth analysis of cloud adoption among banks. For instance legacy systems could be perceived as both a driving force or a challenge opposing the adoption of clouds. Cloud services have clear advantages in terms of security over legacy systems. However migrating from a legacy system to cloud can be a challenge with a poor system design that does not support such migration or perhaps untrained employees that do not have a grasp of modern cloud environments.

The adoption of moving bank services to a cloud has been influenced by many factors. A study done by Ibrahim [20] illustrates the potential facilitators and inhibitors. While by combining mobile devices and cloud computing's ability provides better services and products for clients, and at the same time, could reduce the operation costs. Mobile banking could be easier to accept if it provides certain advantages to clients. For instance, despite its complexity, applying cloud in banking provides user-friendly interfaces and it is easy to use. Compatibility is another one, as it allows clients to access services anytime and anywhere, and the transaction results could be seen without any delay. However, the degree of risk is the main concern of clients, as well as the privacy and security issues. Some clients are even afraid that hackers would hack their accounts and get their PIN code.

3.5 Security and privacy issues

An electronic transaction can happen anywhere and anytime. People use their mobile devices to deal with the transaction. With more people experiencing the convenience of e-commerce, certain issues arise.

Jem-Ho et al. [14] proposes a model, as mentioned in ***Models***, which deals with possible privacy issues and security risks. For service providers, they might meet a malicious client who denies the transaction. Generally, payment gateway, message authentication code (MAC) or symmetric key is used to keep consumers' privacy and the payment information between them to stay unchanged. However, the computation and communication costs might increase if a client keeps many different keys for different merchants. Apart from it, there might be attackers. The proposed model aimed to handle the risks. As anonymous transactions are used and payment information is protected by a one-way hash function, there is no way attackers can get the actual information about the client and the payment. Even if they can, the attack still can not happen as the buying time of the client is also sent to the payment gateway and the receiver could check the timestamp. In summary, the proposed model protects users' privacy and also on the other side, protects merchants' rights by involving a payment proof.

As mentioned in ***Banking in the clouds***, cloud-based banking has also become a popular trend. A major factor that allows banks to switch to cloud

environments is the level of security that is offered by the modern cloud services. Sczesnak[18] identifies 10 cybersecurity practices that help cloud vendors compete against in-house data centers. The most important is the 'Network effect', where the investments from all the customers of the cloud service are combined to build and run a highly secure platform. Among these practices are also experience and in-depth knowledge of how to operate a service in a cloud environment, something that many industries are still struggling to catch up with. Additionally, most cloud service providers have obtained various certifications in security and have trained their employees to specialize in keeping the system protected from malicious attacks.

Qin et al.[11] focus on different challenges that arise from using cloud environments for mobile payments, most importantly in the areas of security and privacy. Not only they identify the potential security threats, but also formalize the security requirements that a cloud payment system should adhere to. These requirements are:

Unforgeability. It is impossible to make a fake payment or submit a fake receipt by impersonating another user in the system.

Anonymity. Personal user data, including user identities, is kept confidential.

Traceability. Payments can be traced using a unique identifier, which makes it impossible to deny the sending or receiving of payments.

Non-repudiation. Merchants cannot deny the correctness of the information on the receipt. Customers cannot deny the correctness of the information about their confirmed payment.

Small overhead. All the computation and processes that take place on the mobile device must be limited to a minimum amount of resources, due to mobile devices having performance and battery efficiency issues.

These requirements are the bare bones of the system, which Qin et al.[11] introduce later in their paper. Their implementation of a secure and private mobile payment system is done through incorporating the certificateless digital signature and pseudo-identity cryptography techniques. Moverover, Liao et al.[16] and Kang et al[17] fixed some flaws in the security protocol proposed by Qin et al.[11] to guarantee the forgetability and tracibility mentioned above.

Madhoun et al[15] introduced that NFC payment using EMV(Europay Mastercard Visa) could be risky, as the transaction could be executed in an open environment and attackers might easily get access to private payment information using a NFC reader. The addressed issue was solved by using a proposed security protocol that was proved to be effective when securing mobile NFC payment transactions.

3.6 Innovative features

Cloud computing has become a mature technology and cloud-based payment is getting popular now. However, we still need to explore their potentials to enable further digital business transformations.

Sczesnak[18] points out how being able to bring new innovative features to market more quickly results in a better performance of the system overall.

Cloud providers facilitate the development of such innovative ideas by keeping their cloud platforms up to date with the modern needs of the market. One of the examples is when a bank switches from an internal collections website to a cloud system, which leads to a major improvement in collections performance. Collections is a complex field that involves numerous debt collecting operations. Nowadays there exist SaaS solutions that offer lower cost, better performance and increased compliance for the collecting operations of any business.

From a cloud computing side, there are still different aspects that worth exploring. A survey [21] shows the next wave of cloud computing will be combination of machine learning techniques, artificial intelligence and also the Internet of things (IoT). Combining these three study areas with cloud computing will bring business competitiveness to the market. Not only it would change the way businesses operate, but also it could improve the interaction between businesses and clients. When applying these new features to cloud-based payment systems, it will lead to a new dimension of innovation and technology.

4 Conclusions

Over the years the evolution of payment systems has brought to the market new cloud-based payment methods that facilitate a number of fast and secure payment services available for the customers to use. It also created a number of challenges that led to new research areas and numerous studies aimed at improving cloud environments and payment systems in general. The goal of this paper is to explore modern payment systems by getting a better understanding of the current state of cloud-based payment, the main trends and research in this area, as well as possible future implications.

This section presents the answers to the previously identified research questions. These research questions are helpful in summarizing the conclusions that have been drawn from the studied literature.

- Q1: What is the current state of cloud-based payment?

Many businesses, including banks have been successfully migrating to the cloud-based payment systems, yet some of these migrations have been very slow due to legacy systems not being able to support such migration. However there are undeniable benefits of switching to a cloud environment.

- Q2: What are the characteristics and current research trends of cloud-based payment?

Some of the characteristics that have been identified are the success of the payment systems, as well as high adoption rates. Also major security improvements, highly valuing privacy and anonymity and introduction of innovative payment methods that facilitate development of new technology.

- Q3: What are the possible future research directions for cloud-based payment?

Future research should include methods to further increase the adoption rates of cloud-based payment systems, perhaps through expanding the target audience of new payment methods.

From a business perspective, a possible implication of this study is that it is usually a correct decision to move to a cloud environment instead of maintaining an outdated legacy system. Another implication is that exploring different areas of application of cloud-based payment systems will result in a better understanding of the current customer needs as well as being able to predict possible future trends on the market.

From a user perspective a possible implication of this study is that it is wrong to assume that cloud environments are less secure than private servers, as it might not hold anymore with the recent developments in Cloud Computing.

4.1 Research gaps

There is some literature that covers various aspects of cloud-based payment systems , however there are quite a few gaps that are not addressed in any of the papers. The first research gap is the simplification of cloud-based payment systems and user-friendliness. It is clear that the further we progress with online payments the more complex it becomes and the more different it gets from the usual payment methods, such as paying by cash. Modern payment systems being too complex is a potential threat to the future developments in the payment systems field.

In the literature limited attention has been given to alternatives of cloud computing when it comes to payments. It is important to remember about the alternative ideas, which in this case are companies setting up and maintaining their own in-house server. A lot of literature focuses on destroying the stereotype of modern cloud environments being more vulnerable to malicious attacks. Other than looking at security it is important to explore further such topics as cost-efficiency, limited features of clouds or network dependency, which may cause a large downtime in the system's functionality. It is also worth exploring the positives of clouds, such as scalability or flexibility.

Moreover, there is a big lack of published papers/online articles for "Definitions" and "Innovative Features" of cloud-based payment systems.

4.2 Future research

Cloud-based payment is an evolving service that will keep changing constantly to meet any new demand on the market. New studies can be conducted about user acceptance of new payment systems or methods to reach out to different user groups and broadening the user base. Second, payment systems need more research into alternative solutions to using cloud environments. More in-depth comparisons will help create a better image of what makes cloud a better choice for all the businesses who still use legacy systems. Even research looking into scalability and flexibility of cloud payment services can help speed up the adoption of newer payment methods.

References

[1] "Cloud-based payment solutions. https://ebizcharge.com/2019/12/16/cloud-based-payment-solutions/."

[2] "The future of payments is cloud – and now is the time to embrace it. https://bankingblog.accenture.com/future-payments-cloud-now-time-embrace."

[3] "Six reasons you need to consider a cloud-based payment system. https://ipsi.com.au/six-reasons-need-consider-cloud-based-payment-system/."

[4] "How to write a literature review. https://www.scribbr.com/dissertation/literature-review/."

[5] J. Webster and R. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly*, vol. 26, 06 2002.

[6] M. Stieninger and D. Nedbal, "Characteristics of cloud computing in the business context: A systematic literature review," *Global Journal of Flexible Systems Management*, vol. 15, pp. 59–68, 03 2014.

[7] L. Novais, J. M. Maqueira, and Ángel Ortiz-Bas, "A systematic literature review of cloud computing use in supply chain integration," *Computers Industrial Engineering*, vol. 129, pp. 296 – 314, 2019.

[8] R. Briner and D. Denyer, *Systematic Review and Evidence Synthesis as a Practice and Scholarship Tool*, pp. 112–129. 01 2012.

[9] "Moving payments to the cloud. https://www.ftni.com/blog/moving-payments-to-the-cloud."

[10] M. Almasri and H. Alshareef, "Mobile cloud-based e-payment systems in saudi arabia: a case study," pp. 5–10, 09 2019.

[11] Z. Qin, J. Sun, A. Wahaballa, W. Zheng, H. Xiong, and Z. Qin, "A secure and privacy-preserving mobile wallet with outsourced verification in cloud computing," *Computer Standards Interfaces*, vol. 54, pp. 55 – 60, 2017. SI: CCSP-SR.

[12] Ni Zeng, Shunxi Li, Zhuo Chen, and Cheng Huang, "Technology road map drawing of cloud-based payment based on bibliometrics approach from mining the patent and literature," pp. 237–243, 2016.

[13] "Guess which digital payment method is most popular. https://www.fool.com/investing/2017/11/01/guess-which-digital-payment-method-is-most-popular.aspx."

[14] J.-H. Yang and P.-Y. Lin, "A mobile payment mechanism with anonymity for cloud computing," *Journal of Systems and Software*, vol. 116, 07 2015.

[15] N. El Madhoun, F. Guenane, and G. Pujolle, "A cloud-based secure authentication protocol for contactless-nfc payment," 10 2015.

[16] "Analysis of a mobile payment protocol with outsourced verification in cloud server and the improvement," *Computer Standards Interfaces*, vol. 56, pp. 101 – 106, 2018.

[17] B. Kang, J. Du, L. Si, and M. Xie, "Analysis and improvement on a mobile payment protocol with outsourced verification in cloud service," *Wuhan University Journal of Natural Sciences*, vol. 24, pp. 223–228, 06 2019.

[18] M. Sczesnak, "When is processing payments in the cloud more secure?," Jan 2018.

[19] W. K. Hon and C. Millard, "Banking in the cloud: Part 1 – banks' use of cloud services," *Computer Law Security Review*, vol. 34, no. 1, pp. 4 – 24, 2018.

[20] "Mobile banking adoption: Application of diffusion of innovation theory," *Journal of Electronic Commerce Research*, vol. 13, pp. 379 – 391, 2012.

[21] "Three trends that define the next phase of cloud computing. https://medium.com/sap-innovation-spotlight/three-trends-that-define-the-next-phase-of-cloud-computing-4782d29d15dd."

5 Contributions

Section	**Contributors**
Abstract	Alex
Introduction	Yizhen & Leyu
Methodology	Leyu
Definitions	All
Popularity and success factors	All
Models	All
Banking in the clouds	All
Security and privacy issues	All
Innovative features	All
Conclusions	Alex

Impact of GDPR on Personal Data in the Cloud

Kailainathan Muthiah Kasinathan
University of Amsterdam
12937827
kailainathan.muthiahkasinathan@student.uva.nl

Haritha Jayaraman
University of Amsterdam
12975052
haritha.jayaraman@student.uva.nl

Richard Bieringa
University of Amsterdam
10691065
richard.bieringa@student.uva.nl

Abstract

Personal data is often subject to threat when it comes to storage and processing in cloud environments. The General Data Protection Regulation was introduced in the European Union and one of its purpose was to bring tighter restrictions with regards to handling personal data in the cloud. This paper aims to provide an ideal understanding of how these rules impact the safety of personal data in the cloud in the European Union(EU). The paper also dives into the details of Cloud Forensics and how it can be averted to an extent without affecting the GDPR.

1 Introduction

Cloud computing is an internet based model that empowers on-demand ease of access to a shared pool of resources over the Internet through a pay-as-you-go approach[1]. These resources are provided by cloud providers through different service models such as Infrastructure as a Service (IaaS), Platform as a Service (Paas) and Software as a Service (SaaS). These models allow users to utilise cloud resources without having the responsibility to maintain or upgrade these resources which allows the user to benefit from the massive economies of scale.

One of the main issue regarding adapting to the cloud infrastructure is the challenge of data security. Security issues start with the type of cloud computing deployment models present. The majority of distributions for cloud services in small and European companies are hybrid (a federation of public, private or partner clouds), partner (owned and managed by a trusted partner), or public (owned and managed by an unrelated business); the smallest portion is private (owned and managed internally) as reported by the European Union Agency for Network and Information Security[2]. The Data Protection Directive (DPD) provides a legal framework for data protection in the European Union. The major change in the DPD was introduced through the Treaty of Lisbon in December 2009 by which the data protection was given the status of fundamental right[3]. The DPD was not equipped to manage the explosion of data we have observed in the recent past due to social networking and high use and demand of cloud computing technology for storage and processing of large amounts of data[4]. Moreover it was not mandatory for all the member states to follow the same standards, which led to some of them adopting only the minimum standards, leaving poor data protection and creating uncertainty for the adoption of cloud computing technology[5]. As per a survey of European companies named Gallup Survey it is revealed that there is wide heterogeneity in understanding and implementing the data privacy rules among the member states[6]. Here is where the need of General Data Protection Regulation (GDPR) regulations come into play. On May 25 May 2018, GDPR regulations entered force [7] overriding the Directive 95/46/EC[14]. The GDPR maintains a balance where both the data controller and the data processor share the responsibilities in case of any

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

form of breach while the latter focuses only on the controller. A comparison of the DPD directive and the changes created by GDPR in [8] gives a clear distinction among the two. In order follow the new GDPR regulations it is mandatory to understand what it means to be a data controller and a data processor and their roles and responsibilities which will be discussed in section 2. One more important thing to know when it comes to GDPR is the data access and data processing. As cloud computing models are involved, it is also mandatory to know the amount of data each of these models contain. Among these models, with respect to data access, IaaS instance provides more information than the other models as this provides the infrastructure where the customers install and set up the image for security analysis purposes and to execute snapshots of them on the virtual machine[2]. With regards to data processing, SaaS and IaaS technologies reside at the extreme of the same scale which makes their providers have different roles and responsibilities[2]. The reason is IaaS providers cannot customize the services as they only provide infrastructure and are completely unaware of its purpose and usage. On the other hand SaaS services provide a wide range of control in relation to the data processed by the customer. This makes SaaS providers to provide the needed security level for each of its services which IaaS cannot because of being unaware of the data involved. All these information can help in providing more secure personal data and also being complaint to the GDPR.

The paper's aim is to give a clear understanding of how the GDPR acts in ensuring the safety of Personal data in the cloud environment. In other words, the research question is "Is Personal data safe in cloud with the new GDPR regulations?".The paper is structured as follows: Section 2 deals with the background details which includes the actors involved in the the Cloud Environment before and after GDPR and also explains the personal data life cycle. Section 3 discusses the new provisions in GDPR and a detailed explanation of the important characters involved followed by a fictitious example. The loopholes of GDPR is discussed in Section 4 and a daunting challenge named Cloud Forensics in Section 5 along with recommendations to overcome the problem. We then discuss the future work in Section 6 and wind up with the conclusion in Section 7.

2 Background

2.1 Actors in the Cloud Environment

- Data Subject: A Data Subject refers to any person whose data is collected, held or processed and whose consent plays a very important role in the GDPR. The data subject should also be given the right to access the data.
- Data controller: A Data controller is the one who controls the procedure and purpose of data usage. The data gathered is either processed directly by the controller or the controller can opt to use a third party to do it on their behalf. This does not give the third party control over the data, the data controller remains the party which decides the amount of data that needs to be processed. In short, the controller acts as a dictator on how and why data is going to be used[9].
- Data Processor: Once the data to process is decided by the controller the processor processes them. They do not own the data which does not allow them to change the purpose and means of the data used and they are bound to follow the instructions of the data controller. The process is the third party the controller assigns to process the data[9].
- Data Protection Officer: The primary role of a Data Protection Officer (DPO) is to check if the organisation in which she/he is working for processes the data subject's personal data as per the regulations. According to the Regulation (EU) 2018/1725 , the Data Protection Regulation, the EU institutions and bodies are obligated to appoint a DPO. And According to Regulation (EU) 2016/679, it obliges some organizations in the EU countries to appoint a DPO, which is applicable from 25 May 2018[15]. The DPO officer is also required to possess expert domain knowledge on data protection rules and also an understanding of the organization so the officer can also help in providing advice and recommendation in interpreting the rules and implementing them. Some of the tasks include to ensure that controllers and data subjects are informed about the data protection rights and the obligations and responsibilities. They should increase awareness about these rules and in case of failure to comply bring them to immediate attention to the EU institution. The DPO also ensures that the EU institution is accountable to the data protection rules compliance. Any queries or complaints prevailing in the institution are handled by the DPO on her/his own initiative

or when a data controller or any other person from the institution requests for it. The DPO also cooperates with European Data Protection Supervisor (EDPS)[15].

- Data Protection Authority The application of the data protection law by the EU Institutions and bodies are supervised by the Data Protection Authority (DPA). They are independent authorities who supervise through their investigative and corrective powers. There is one DPA for each Member State and they are the main point of contact with any question on data protection[16].

2.2 Personal Data & Life cycle

The GDPR defines Personal data as "any information relating to an identified or identifiable natural person".[10]The main objective of the GDPR is to increase the data subject right in terms of dealing with personal data. The data subject should be more in control of how his/her personal data is processed. The personal information life cycle as seen in figure 1 consists of the following stages as defined in [22]: Collection, Storage, Disclosure and Distribution, Use and Retention, and finally destruction.

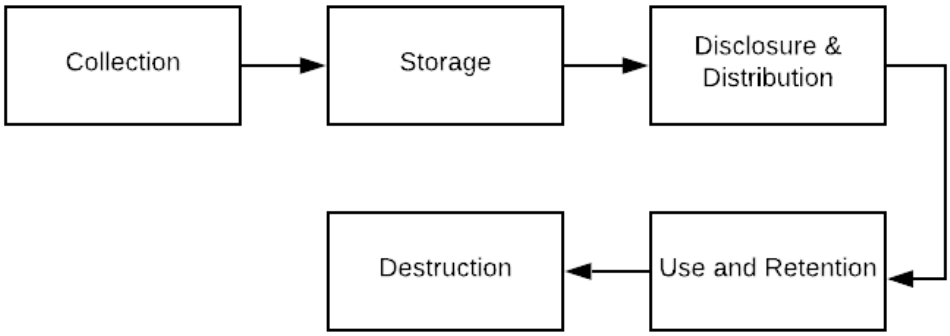


Figure 1: Personal Data Life cycle in Cloud

3 GDPR in Cloud Computing

GDPR has a number of significant changes in the area of cloud computing. It brings in a number of changes in the way personal data is collected and Processed. As referenced in section 2, the two major roles - Data controller and Data Processor in the cloud are given new responsibilities with the introduction of the GDPR.This section gives the new changes in the rules and gives a basic understanding of the storage and processing of personal data in the cloud. The essence of GDPR when it comes to cloud computing is the new way of looking at it from both the Data controller and the data processor's perspective. The previous directive was fully focused on the Controller that lead to a lot of challenges when it came to processing the personal data.

New provisions in GDPR: The provisions are listed according to [8]

- An Uniform approach to Data Protection Laws in EU: In this provision, it states that all member states in the EU follow a single set of rules on data protection invalidating the current DPD. By doing so the standards are improved and perplexity among the member states are avoided.
- Processor fully Accountable: In the DPD, the controller is solely responsible but in the new provisions of GDPR both data controller and data processor. By doing so, in the GDPR, the processor is directly accountable for data processing although they process data for data controllers and subject to data protection and security rules.

- Applicable to Non-EU Companies: In the new GDPR regulation all providers who process personal data for the EU resident are accountable and should abide by the GDPR rule. This means that any provider in and outside of the EU extending the territorial scope of the EU data protection law.
- Appointment of Data Protection Officer(DPO): The law requires an appointment of DPO by the controller and the data processor where more than 5000 data subjects are being processed for more than 12 months in a period[11].The DPO acts as a data processor that does not exists in the EU.
- Data Breach Notification: According to GDPR, any form of personal data breach should be reported within 72 hours to the authority once the organization is aware of it. The Article 33(3) [12] states the four requirements when reporting such a breach.
- Right to Data Portability: With the GDPR Regulation the data subject is provided the right to request to move its personal data stored in one controller to another controller. This comes under the Article 20 of the law.
- ONE-STOP Initiative: This introduces one Data Protection Agency for all EU residents and different data processor and controllers[12]. This means that irrespective of the organization, all of them in each member state will have a DPA in the established country. This also creates uniformity among all the member states of the European Union.
- Serious Penalties for Negligence Breach: When a data breach happens due to a negligence and leads to data and privacy loss the law proposes a fine up to 5% of the annual revenue and maximum up to 100 million euros[13]. Unjust Enrichment[8] is a potential offence which happens when a company has saved money and does not apply adequate security measures which can lead to serious consequences to the data breach.
- Right to be forgotten: The right to be forgotten states that if any EU resident no longer want their data to be processed or stored, they can request for the complete removal of data.

3.1 Data Processors and their Importance

GDPR states that the term Processing is "any operation or set of operation that is performed on personal data or sets of personal data, whether or not by automated means"[2]. The GDPR also states that processing of “personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership,” as well as “genetic data, bio-metric data for the purpose of uniquely identifying a natural person, data concerning health, or data concerning a natural person’s sex life or sexual orientation" is prohibited[2]. These considerations make the data processors role very important when it comes to handling personal data. This is the reason why GDPR details the rules for Data processors and also lays down in-detail explanation to sub contracted data processors and also joint controllers. Furthermore, in the event of a security issue the data processor is more suited to handle that than the controller which further reiterates the importance of data controllers.

3.2 What does GDPR mean to data processor and controller?

The GDPR keeps the controllers rules similar to the ones found in DPD. Since data processors are also included, there are many changes in the way the personal data is handled in the cloud[23]. As per the law, the processors are bound to meet the GDPR requirements by taking appropriate measures to ensure the protection of the rights of the data subject as enlisted in the GDPR. The same applies for the data controller. The controller should also select a processor who satisfies all the criteria outlined by the GDPR. The essential thing to be noted is that at no point the activity of the controller or the processor should turn out to compromise the Data subject’s rights as prescribed in the GDPR.

Many processors tend to outsource their work which is not an unusual sight in cloud computing. This process of outsourcing is done to maintain constant levels of processing data at all periods of time. Also keeping in mind, increase in data in organizations tends to increase the complexity of the cloud processes. With the new rules, Processors cannot outsource or engage any other processor without authorization of the data controller which gives the necessary transparency and accountability into how the personal data is processed. By getting the controller’s approval, the controller is now aware of who the processors make use of to carry out their tasks. The controller and processor can enter

into a contract which specifically states what data is used, the main aspects of processing and other obligations. Adding on to the outsourcing context, the GDPR also clearly states that if the processor's subcontractor fails to meet the requirements then the initial processor is fully liable to the controller.

To provide an appropriate level of security, the processor and the controller are bound to assess the risks of processing the personal data. It should be carried out keeping in mind the cost of implementation and the nature of personal data being processed.Suitable measures to mitigate risks should be in place. GDPR formulates certain appropriate measures with respect to personal data which are as follows:

- Pseudonymization and encryption of all personal data.
- Ability to ensure confidentiality, integrity, availability and resilience of processing systems.
- Ability to restore the access to personal data in case of an incident.
- Process to regularly test,assess and evaluate the measures taken towards the security of processing.

Adding on to this, the processor is liable for any damage caused to the data subject only when the processing did not comply with the rules set out by GDPR. In such case the processor will also be held accountable to the compensation. The data processor is also responsible for reporting any breach of personal data to the controller. There should be no delay in communicating the information about breaches where the GDPR sets the threshold at 72 hours since the time of the initial data breach. The processor is also subjected to comply with the obligations of the controller and also it should be available for any audit the controller performs to verify the security of the processing of personal data.

The GDPR also gives a clear understanding about the geographical boundaries where it will be applicable. As per the article, GDPR is applicable to any processor or controller that uses the personal data of an European citizen. This belongs to processors and controllers outside the European union too. Business models often lead to joint controller scenarios.Thus GDPR states that,the joint controllers must specifically state their roles,relationship and responsibilities between them. The final aspect of GDPR which aims towards securing personal data is on cross border data transfer. Data can be transferred to any country, region or territory which is tagged by EU Commission to have adequate data protection capabilities. If the tag is not present, transfer can happen following corporate rules and regulations. These are the major changes brought about by GDPR with respect to personal data in the cloud. A proper framework has been established with respect to how data is processed which is the success of GDPR.

3.3 A Fictitious Example

To understand the implementation of the rules we actually came up with a fictitious example as depicted in figure 2. We have referenced a part of the work done by P.T.J Wolters[17] in building this example and have limited the scope to explain this in relation to the data subject, data processor and the data controller. To understand the real world situation, we take a sports firm which is based out of France. The firm basically collects a lot of personal data of its customers. The data ranges from billing information, addresses of customers to contact information and preferences etc. Here the customers are the data subjects. The Sports firm has customer service operations throughout all of Europe. These customer service operations are independent companies where the firm has an outsourcing contract agreement with. The operations are provided by a specific Dutch company which handles the operations within The Netherlands. The personal data collected is stored in the cloud which is provided by a company operating out of Belgium. All the involved companies are based in Europe and handle the data of European citizens which means it is essential for them to be GDPR compliant. With this example we can see that the Sports Firm is the data controller and the companies carrying the customer service operations are the data processors.

The sports firm being the controller decides the purpose of the data and the means of processing the data collected. The main responsibility of protecting the personal data lies with the controller. As per the rules, the sports firm chooses the customer service provider companies and the company which acts as the storage provider after carefully assessing whether they will be able to satisfy the requirements in terms of protecting the personal data. These data requirements are settled in a contract between the controller and the processors. The processors, which consist of the customer service

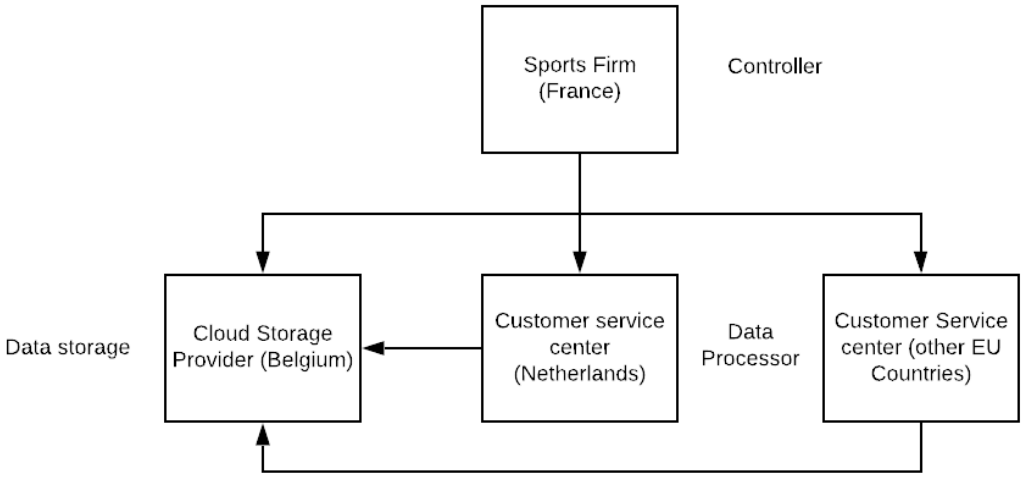


Figure 2: Fictional Example - Tree

companies, generate, handle and use the personal data to provide their services. The data collected by these companies are made available to the sports firm by storing it in the cloud. All the processing is done on behalf of the controller. By the rules according to the GDPR, the processor is also required to have the right measures in place to ensure the protection of the personal data. In case of a data breach the customer service companies are considered to be fully liable, this also applies for the cloud storage provider. GDPR considers them as processor since they make the data available to the Sports firm as well as the customer service companies. Thus this example provides a good understanding of how the data controllers and processors are responsible in protecting the personal data of the data subject involved.

4 Loopholes of GDPR

Although the GDPR has brought a significant amount of positives towards protecting the personal data, the regulation has some loopholes. Ihenayi Samuel has outlined some of the important missing links in the GDPR in his research[18]. As per Samuel, there are few loopholes present in the GDPR laws related to personal data in cloud. These are stated as follows:

- The GDPR implements a strict division between the data processor and data controller which is misleading as there are more actors and layers present within a cloud computing framework.
- Cloud computing is a place where there are a lot of collaborating, but autonomous entities present. Describing their relationship in terms of principal-delegate or as a relationship of command makes no sense. This can lead to legal uncertainty in understanding the roles of the actors within the data processing chain.
- Retaining the use of modern contractual clauses is another mistake as they do not cover all aspects of cloud transactions. Some of the clauses of the model do not fit the technical and organizational frameworks of cloud services. Asking the data processor to submit for audit when requested by the controller is not a feasible operation. It is also impossible to get the consent of the customer before engaging in all the sub-processing services.
- The inability to transfer data between two different processors or controllers who both have approved binding corporate rules but do not belong to the same group is illogical. This contradicts the way where two third countries who have adequacy status are allowed to transfer EU data between them.

Samuel states that these problems arise due to the missing knowledge of the cloud architectures, features and business models by the lawmakers. They also base this on general assumptions of laws that do not always apply or translate well to the cloud.

5 Cloud Forensic Problem - A big challenge to GDPR

The cloud systems are constantly under serious attack from intruders trying to get hold of personal data. Once an attacker gets into a cloud system and becomes an intruder, there is very little anyone can do to protect the data covered under GDPR. The intruder can access, modify, delete or extract personal data from the cloud system which is considered a serious breach. The resulting data can be abused in numerous ways. This is a serious issue and often referred to as "The elephant in the room" in cloud circles. This is a problem which is known to everyone but no one is ready to discuss it and the problem is tough to defend against. This makes it even more complex to find solutions to overcome the problem. The complexity of the challenge makes it a problem for organisations using the cloud and this also proves to be an obstacle to attaining compliance to GDPR.

5.1 Definition

Cloud Forensics is defined as "the application of computer forensic principles and procedures in a cloud computing environment"[20]. According to Ruan et al.[21], cloud forensics is defined as a subset of network forensics. When an intruder comes in and gets hold of the data, in most cases the intruder leaves very little forensic trail to follow which makes companies to be unaware of any breach happening. When they are not even aware of the breach, they will not be aware of the records deleted, accessed, modified or stolen. There are some significant cloud forensic challenges which makes it a complex problem to solve in cloud computing. These challenges can be seen in figure 3 below, which gives a brief overview of the challenges faced in cloud forensics according to the different cloud platforms.

Challenges of Cloud Forensics	Exists in		
	IaaS	PaaS	SaaS
Physical inaccessibility	✓	✓	✓
Dependence on CSP	✓	✓	✓
Volatile Data	✓	×	×
Trust Issue	✓	✓	✓
Large bandwidth	✓	×	×
Multi-tenancy	✓	✓	×
Decentralization of Logs	✓	✓	✓
Volatility of logs	✓	×	×
Logs in multiple tiers and layers	✓	✓	✓
Accessibility of logs	✓	✓	✓
Depending on CSP for logs	✓	✓	✓
Absence of critical information in logs	✓	✓	✓
Chain of Custody	✓	✓	✓
Problem of current forensic tools	✓	✓	✓
Crime scene reconstruction	✓	✓	×
Cross border law	✓	✓	✓
Presentation	✓	✓	✓
Compliance issue	✓	✓	×

Figure 3: Challenges in Cloud Forensics[20]

5.2 Personal data and GDPR Compliance

The cloud forensic problem poses serious threat to attaining GDPR compliance and if that is not met then your personal data is at risk. It is impossible to be GDPR compliant by ignoring the cloud

forensic problem. As seen from the previous sections, it is evident that there is nothing to stop the intruder once he gains entry into the cloud system. The regulation under scan here is the one where GDPR states that "data breaches should be reported within 72 hours of breach and all information about the data compromised should be made available". This is where the compliance will come into question as most of the companies are not aware about the breach happening or what data is getting compromised. Also under GDPR personal data should be encrypted. If the data is yet to be encrypted and if the encryption and decryption keys are under the same cloud instance, in the event of an intrusion a company will not be able to provide any information about whose data was compromised. This will lead to multiple breaches of the GDPR provisions. Author Duncan outlines ways to achieve minimum compliance with GDPR by keeping the cloud forensic problem in mind. He gives solutions to attain compliance of GDPR which are: Right of Access, Right to Erasure, Privacy by design, reporting a data breach and finally notifying the data subject in the case of their data not being encrypted. As per Duncan[19], to achieve minimum compliance organizations should follow the recommended methods which are as cited in his research as follows:

- All personal Data should be encrypted, the process of encryption should be performed locally while the encryption and decryption keeps should not be stored in the cloud environment.
- Off-site maintenance with a full audit trail of the organization's database.
- Off-site maintenance of forensic records of all users accessing and performing activities on the database must be carried out.

These will help in reaching compliance and keep the organization from breaching the rules of GDPR and also protecting the personal data of the data subject.

6 Future Work

Cloud computing is a field where advancements in terms of architecture, technology used and functionalities provided are ever present. One of the key problems of the GDPR is the ability in which the laws framed fail to take in the practical aspect of cloud computing. Some of the GDPR laws towards personal data protection cannot be applied in a real life environment. Thus as a recommendation towards improving this aspect, it will be good if renowned experts in cloud computing are kept in the discussions when framing the law. Also as we discussed the cloud forensic is a major issue. There are some methods out there to solve the intruder identification problem which we have but they are not that effective. More dedicated research can be done towards solving the cloud forensic problem instead of avoiding to work around it as it tends to cost companies a lot of money and mistrust with data subjects.

7 Conclusion

It is evident from GDPR that it is a definitive framework when it comes to securing the personal data in the cloud, which was not the case with the old DPD directive. The GDPR has given the data subject more rights, brought the data processor into the picture and handed them more responsibilities when it comes to handling personal data. This gives a clear idea that the GDPR has strengthened the personal data privacy in the cloud but only to an extent.There are still some uncertainties in the way GDPR handles some of the regulations as we saw in the loopholes section and also the ability of GDPR to handle the cloud forensic problem is still a question mark with both of them not suiting each other. Thus with this research we intend to conclude that the protection of personal data in the cloud has been strengthened by the GDPR but there are still problems which can pose a threat to the safety of personal data.

References

[1] Hussam Alddin Shihab Ahmed & Mohamad Fadli Bin Zolkipli, Data security issues in cloud computing: review *International Journal of Software Engineering & Computer Systems (IJSECS)*ISSN: 22898522, Volume 2, pp. 5865, February 2016.

[2]Barbara Russo, Laura Valle, Guido Bonzagni, Davide Locatello, Marta Pancaldi & Davide Tosi, Cloud Computing and the New EU General Data Protection Regulation*IEEE Cloud Computing;* https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8552651

[3] Treaty of Lisbon amending the Treaty on European Union and the Treaty establishing the European Community [2007] OJ C306/01.

[4] See Commission, 'Data Protection' (2015) Special Eurobarometer 431/ Wave EB83.1 – TNS opinion & social, Summary, 2 https://ec.europa.eu/commfrontoffice/publicopinion/archives/ebs/ebs_431_en.pdf accessed 10 May 2016.

[5]https://ec.europa.eu/commfrontoffice/publicopinion/archives/ebs/ebs_431_en.pdf

[6] Nir Kshetri & San Murugesan, Cloud Computing and EU Data Privacy Regulations ; https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6489955

[7] European Regulation (EU) 2016/679 of the European Parliament and of the Council, "General Data Protection Regulation," 2016.; Available: http://data.europa.eu/eli/reg/2016/679/oj

[8] Sohail Razi Khan & Professor Luis Borges Gouvia, The implication and challenges of GDPR's on Cloud Computing Industry *IPASJ International Journal of Computer Science (IIJCS)* ISSN 2321-5992 Volume 5, Issue 7, July 2017

[9]https://digitalguardian.com/blog/data-controller-vs-data-processor-whats-difference

[10] Art. 4 GDPR Definitions, https://gdpr.eu/article-4-definitions/

[11]See also Dan Jerker B. Svantesson, The Extraterritoriality of EU Data Privacy Law—Its Theoretical Justification and Its Practical Effect on U.S. Businesses, 50 STAN. J. INT'L L. 53, 65, 73, 100 (2014).

[12]David Loshin, Seven Data Strategies for Regulatory Compliance; http//governyourdata.com/page/white-paper

[13] In this regard, see M Brkan, 'The Unstoppable Expansion of EU Fundamental Right to Data Protection. Little Shop of Horrors?' (2016) 23(5) Maastricht Journal of European and Comparative Law 812

[14]Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, 1995 O.J. (L 281/31) ("Data Protection Directive"). IDC 2012, p. 48 – 64.

[15]https://edps.europa.eu/data-protection/data-protection/reference-library/data-protection-officer-dpo_en

[16]https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-are-data-protection-authorities-dpas_en

[17]P.T.J.Wolters, The security of personal data under the GDPR: a harmonized duty or a shared responsibility?,*International Data Privacy Law, Volume 7, Issue 3, August 2017;* https://academic.oup.com/idpl/article-abstract/7/3/165/3860950?redirectedFrom=fulltext

[18]Iheanyi Samuel Nwankwo, Missing Links in the Proposed EU Data Protection Regulation and Cloud Computing Scenarios: A Brief Overview, 5 (2014) JIPITEC 32; https://www.jipitec.eu/issues/jipitec-5-1-2014/3905

[19]Bob Duncan,"Can EU General Data Protection Regulation Compliance be Achieved When Using Cloud Computing?",*CLOUD COMPUTING 2018 : The Ninth International Conference on Cloud Computing,GRIDs, and Virtualization;* https://abdn.pure.elsevier.com/en/publications/can-eu-general-data-protection-regulation-compliance-be-achieved-

[20]Shams Zawoad,Ragib Hasan,Cloud Forensics: A Meta-Study of Challenges, Approaches, and Open Problems, https://arxiv.org/abs/1302.6312

[21]K. Ruan, J. Carthy, T. Kechadi, and M. Crosbie, "Cloud forensics: An overview," in proceedings of the 7th IFIP International Conference on Digital Forensics, 2011.

[22]Alaa Altorbaq,Fredrik Blix,Stina Sörman, Data Subject Rights in the Cloud,*The 12th International Conference for Internet Technology and Secured Transactions (ICITST-2017);* https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8356406

[23]Rossana Ducato,Cloud computing for s-Health and the data protection challengehttps://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7580803

8 Contribution

Name	Part Of the Report	Contribution
Haritha Jayaraman	• Introduction • Actors in the cloud environment • New Provisions in GDPR • Conclusion	Wrote about the topics mentioned. Researched and gathered references related to these articles and formulated the findings in the report.
Kailainathan Muthiah Kasinathan	• Personal data & Lifecycle • GDPR in cloud computing (Excl. New provisions in GDPR) • Data processors and their importance • What does GDPR mean to the data processor and controller? • Fictitious Example • Loopholes of GDPR • Cloud Forensic Problem - A big challenge to GDPR • Future Work • Conclusion	Wrote about the topics mentioned. Researched and gathered references related to these articles and formulated the findings in the report.
Richard Bieringa	Correction and Validation	Improvised the reports overall look and feel. Provided assistance in correcting the grammatical errors and adherence to the Format and also validated the contents of the report. Contributed a lot towards the initial three assignments of the course, so teammates eased out the burden on literature study.

Serverless Computing and Serverless Research

Kai Zhang SN: 12712469
Tianyang Lu SN: 12971502
Hatim Alsyahani SN: 13281437

Abstract

Serverless computing as a newly merged cloud computing technology has many definition with common ideas also with different opinions at the same time. In this paper we will discuss definitions, characteristics, usages, trending from different papers and also with our own findings. Furthermore, there are a lot of experiments on public serverless cloud computing but hardly to find private cloud deployment, code tests and evaluation, we will do experiments on three different local-based serverless computing technologies (Apache Openwhisk, Knative, OpenFass) and do evaluation for new coming developers or academic users in order to let them choose proper platform for their projects.

KEYWORDS: Serverless computing, FaaS, Apache Openwhisk, Knative, OpenFass, FaaS introduction

1 Introduction

Serverless computing (SC) has gained traction over the recent past due to its many benefits. After the SC applied in the cloud computing field, there's a new cloud services structure called Function as a Service (FaaS). As [**1**]'s *Figure 1(left)* shows that the server-less related popularity boosted since Amazon took AWS Lambda, which is the first FaaS platform, to the market in 2016. They are a kind of future for Cloud Computing.

The first interesting thing we found is that though SC and FaaS are popular for programmers and a lot of literature are discussing SC and FaaS, not only among developers but also among big companies, there's no standard definition on it. Besides, characteristics, architecture, use cases and trends toward SC and FaaS have minor differences among different groups or papers.

Lack of self-deployed framework evaluation is another problem we encountered in paper searching. When we went through lots of papers about implemented technology part, only public commercial cloud technology like AWS Lambda, Azure Functions, Google cloud functions, etc. are evaluated and discussed. For open sourced or self-deployed frameworks, only OpenLambda is introduced, other popular platforms with same popularity with OpenLambda, which can be checked from *Figure 1(right)*, like Knative, OpenFaaS and Apache Openwhisk are totally ignored, no matter to say that there's no introduction, test and evaluation on different platform. This is mainly because most of open sourced frame are developed end of 2017, then comes popular after 2018.[**16**]

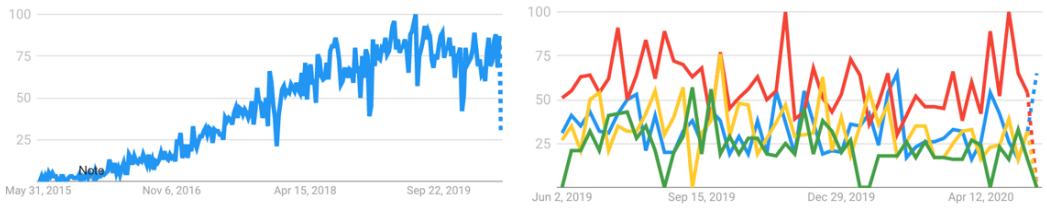


Figure 1: Serverless google trending(left) Self-deployed FaaS google trending(right)

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

In this paper will be spliced into two parts. Section 2 (1st part) is targeted at giving new cloud users a brief idea on what is the SC and FaaS also about its structure. Then what are their characteristics will be discussed, including their pros and cons. Last part of this section will be their difference compared to the other cloud computing structures. The final subsection is use cases and trending. In Section 3 we will give an introduction, implement High I/O & High computing tests, and give evaluation on three k8s based serverless framework[**18**]: Knative, OpenFaaS and Apache Openwhisk, so that can give new developers an idea which platform are suitable for them in implement their own structures and why.

2 Server-less computing and FaaS

2.1 Definition and Architecture

What is serverless computing? From [**1**][**4**][**5**][**6**], we found that it is kind of a new term of industry to define a new development trend.

First of all we would like to clarify a misunderstanding, which are also mentioned in [**1**][**2**][**3**], the SC is never means no servers for computation, but means developers leave infrastructure configurations and operational concerns such as resource provisioning, monitoring, maintenance, scalability, and fault-tolerance to the cloud provider.

About SC definition, from model aspect it defined as: building and running applications 69 that do not require server management, from cost aspect[**13**] aspect it defined as: costs you nothing to run if nobody is using it (excluding data storage), from function level [**14**] and also definition compared to other platform [**3**]: similar to Platform as a Service (PaaS) but a functional level. (Actually, this one is not that exact, because the SC's services can be scaled to 0 when it is not used, but PaaS cannot do that.) As a developer, I think the following definition is the most suitable[**2**]: SC is 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. This tells developers that the SC is elastic, pay-as-customer-consumed, and there's no concern to maintain services, moreover, these are SC's main characteristics.

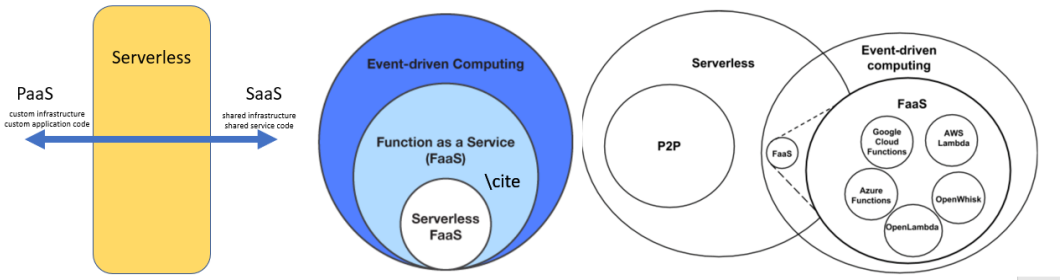


Figure 2: Relation between serverless computing and Faas(left)(middle)(right)

Then what is FaaS and its relation to SC? [**1**] *Figure 2(left)* thinks FaaS is a cloud computing structure between Service as a Service (shared infrastructure and code) and PaaS (custom infrastructure and code), which does not shows how its relation to SC and definition is quite vague. The good point is told us, FaaS is kind of service that use custom code but shared in-structure and can auto-scale resources on demand. AWS and [**5**] *Figure 2(middle)* regard SC is a part of FaaS, and FaaS is a part of Event Driven. Though this is a AWS Lambda start milestone and Leader of FaaS, but this definition has bias, they ignored peer-to-peer service of SC also have big influence around the world [**11**][**12**]. [**2**][**5**][**6**][**7**] *Figure 2(right)* regard SC as a model and FaaS is the most prominent implementation, where deployed services are executed in response to triggers such as events or HTTP requests.

When we comes to talk about SC architecture, there are many difference among SC implementations, however all of them have almost the same high-level platform architecture, we combined the discussion in [**1**][**4**][**5**] to make a new figure in order to illustrate the architecture.(*Figure 3*)

The edge part defines different *triggers*, which triggered events and events are processed by the *event controller*(EC). The EC validates the event to make sure it has appropriate authentication and authorization to execute. Then EC gets the corresponding *container runtime* and sends the event to

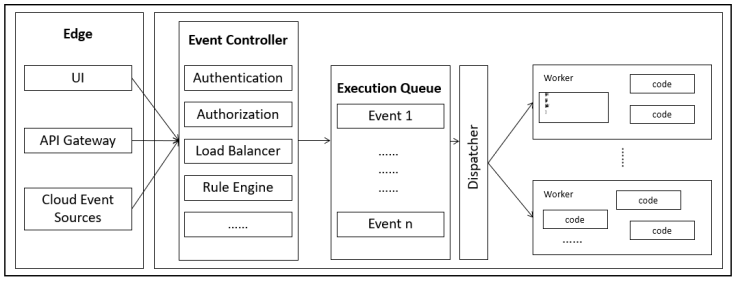


Figure 3: Serverless Computing High Level Architecture

the *event queue* (EQ). The EQ is targeted at ensuring function performance even under heavy system load. Then the dispatcher sends the event in the EQ to a *worker*. The worker executes the function and waits for a response, gathers running logs, returns response to users, and stops the function if it is no longer needed.

2.2 Characteristics

In this part we will discuss SC's characteristics so that we can know its pros and cons, what is its difference compared to the other models (like Iaas, PaaS), and what is the best usages and why.

Besides three characteristics: 1) elastic, 2) pay-as-customer-consumed 3) no concern to maintain services we discussed best definition for developers, the SC also has other characteristics listed as follows:

- Atomic code/Micro-service, small pieces of code are easier to start, interrupt and restart, which also means easier to manage and scale in the cloud. [**2**][**6**]
- Stateless service, a stateful service requires persistent storage, however in implementation there are no auto scaling databases that can scale to 0. [**2**][**6**]
- Limit time on execution, longer time on execution and frequent access is not cost-effective for customers and also harder for service providers to maintain and scale. [**2**][**6**]
- On-demand, the SC executes, scale, and bill functions on users' demand, which can bill and run instances can even scale to 0 when the service is not needed. [**6**]
- Event triggered, the function is executed by specific triggers. For example, client HTTP requests, events by external systems, incoming data streams, or self-defined rules, etc. [**6**]
- Limited Supported Languages, the type available on SC or FaaS depends on the corresponding provider, most providers support Java, JavaScripts, Python. [**6**]

The above characteristics are discussed from formal definition, we also found one outstanding view from industry survey[**5**]:

Stateful service is available, this is not in conflict to the characteristics listed above. The above means that all procedures and resources should be serverless, and [**5**] focus more on function, focus more on service itself, is serverless, and can use external entities.

Due to these characteristics of SC and FaaS brings many benefits for programmers.

- SC is the most cost-effective model, no matter what else elastic models we use there are always a lowest cost to maintain the service(s).
- quality-of-service (QoS) monitoring, scaling, and fault-tolerance properties. At the same time, it gives programmers more availability to think about business logic and related software structure and modules.

However as a new merged technology, it also has some cons and limitations.

- Language, version, library supporting problems. if a developer's application needs a special language or certain version of language or library that is not supported by SC, then it cannot

be deployed, which is annoying. For example, right now python2 is no longer maintained, so providers removed support of python2, then the services needed for python2 is no longer available.

- Cold start, (for providers, developer needs to know) cold start problem, on-demand characteristic can lead to service scale to zero, this can make function cold start. To provide high QoS, the provider needs to solve this problem.
- Scaling, (for providers, developers need to know) load and provision predicting problems. Load prediction is hard but essential for SC providers. As SC function is cold started, and a related amount of provider's resource Is loaded from time to time, providers need to proactively provision resources and predict future requirements in order to make service good response time.

2.3 FaaS Compare to other Cloud Structures

In this part we will discuss Faas's comparison to Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), Mobile Backend as-a-Service (MBaaS), and made a table(*Table 1*) for look their difference. We will gather key points from [**1**][**2**][**4**].

IaaS enables developers to have control on infrastructure is the fundamental difference compared to FaaS. These characteristics enable IaaS users to set up their own running environment, and have great flexibility and the ability to customize every aspect of the application and infrastructure.

PaaS, as mentioned in definition, there's a definition: FaaS is "function-level" Paas. They must have many common attributes, for developers, both of them enable developers not concern on infrastructure configuration and maintaining, and servers can be scaled on incoming demand. On difference, PaaS cannot be scaled to 0, no cold start, more suitable for stateful service and run computation on system level, on the other hand, FaaS can be scaled to 0, so not need to pay for idle resources, target at state-less and micro-service on functions levels.

SaaS, also known as cloud application services. It delivers an application as cloud services, like Dropbox, Google Apps, a cloud provider also provides whole software and data maintenance.

MBaaS, has a close resemblance to FaaS, both of them run codes on demand on the server side without managing servers. However, it was typically limited to mobile use cases.

_	**FaaS**	**IaaS**	**PaaS**	**SaaS**	**MBaaS**
Expertise required	Low	High	Medium	Lowest	Low
Scaling	Auto-scaling	Expert to tune	Auto-scaling	N/A	N/A
Infrastructure Control	Low	High	Medium	Lowest	Low
Deployed work Level	Function	Project	Project	None	Function
Service target	ALL Devices	ALL Devices	ALL Devices	ALL Devices	Mobile

Table 1: Different Cloud Structure Comparison

2.4 Status in industry

Right now we have a general cognition on what the SC and FaaS is, what their characteristics include pros and cons, and the difference they make compared to other cloud models. So what is the current status of industry, what does a developer need to learn and know if a developer wants to engage in? We get info from [**6**]'s survey.

From the survey more than 74% of developers are 5+ experienced IT professionals, in contrast more than half of them's cloud experience is less than 2 years. This phenomenon indicates it is a cutting-edge technology. To learn this, solid programming skill and problem analysis ability may be required, because related technology questions are not widely discussed.

When you want to choose a platform to start learning, which one is the best? The answer is Amazon Lambda (AL), the only one platform in the first echelon, which nearly all of developers (86%) have experience on. The second echelon includes Microsoft Azure, Google Cloud, Digital Ocean, Heroku. These platforms all have around 30% of developers have experience on them.

Then what's the dominant programming language and other preferred skills in industry? The top three programming languages is JavaScript (71%), Python (49%) and Java (30%) and related knowledge are Database services ((e.g., Cloudant, ElephantSQL), Logging services (e.g., Loggly, AWS Logging, ...) and Analytics services(e.g., Spark, Hadoop, ...).

Last but not the least, as this survey[**6**] is conducted in mid 2018, and now it is mid 2020, self-deployed/open-source FaaS platform also had long term development. We will discuss and introduce the top 3 popular candidates in section3.

Right now if you want a project to start with, you can check section2 FaaS usage section, it will introduce most usage these days and will inspire your ideas.

2.5 Use cases

Not surprisingly, due to a list of advantages, Serverless computing is acting an important role in the rapid-developing digital society. A most basic example applying SF which is mentioned in the papers[**1**][**2**] is that a simple image-processing event handler function. It is linked to a data store like Amazon S3. Though its function is thoroughly stateless and idempotent, it can be extended by combining serverless functions to develop more complicated application mentioned below in this section.

In 2017, Barga's keynote mentioned 5 common class of use cases for SF which has been constructed in Industry and Research: web applications, backends including IOT, Chatbots, IT Automation and Bigdata.

2.5.1 Web applications

Nowadays, the framework of applications can be established without the detailing of servers. Instead, cloud provider manages the servers and its provisioning and allocation. An example of static websites is Expedia. Applications can conduct integration of events for their CI/CD platforms, infrastructure governance and autoscaling by using SF and run in a stateless, ephemeral and event-triggered containers. [**2**]

2.5.2 Backends including IoT

A serverless backend shifts backend functions like data and authentication to the cloud which is relatedly different form previously development. By using such technology, developers can pick and integrate the functions they need into their projects. With the advantage of timesaving and less expense-cost, developers can focus on what they want to do. IoT (Internet of Things) is an industry which adopted serverless computing early due to tough issues of scalability and availability and highly variable loads. WeatherGods, a mobile weather app, uses serverless computing as its backend[**2**]. Other examples are IRobot, Aegex, Abilisense, which effectively reduce their expenses by using serveless computing. things like finding recipes and directing audibly step-by-step, reading a Kindle book, achieving movie showtimes or sports schedules.[**3**][**5**]

2.5.3 Chatbots

Chatbots like Amazon Alexa are another type of example. A chatbot is a software application used to conduct an online chat conversation via text or text-to-speech instead of chatting directly with a live human agent. Recently, chatbots are used more and more frequently to save labor costs and customers may save waiting time to solve some simple and frequent issues. Amazon Alexa, a highly interactive AI program, can optimize daily life and make it more convenient with customizable services and connect other household devices. With a device that integrates the voice technology, things like finding recipes and directing audibly step-by-step, reading a book, achieving movie showtimes or sports schedules can be conducted.

2.5.4 IT Automation

IT automation is the process of creating software and systems to replace repeatable processes and reduce manual intervention.[**17**] With AWS Lambda(a kind of SF), IT automation is realized easily. After uploading the code without managing or providing servers, it will be handled by servers and

developers just need to pay for the compute used. Amazon Aurora is a relational database service that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Compared with MySQL, its performance in throughput can be 5 times higher.

2.5.5 Bigdata

In recent years, the amount of data generated by brands has been increased dramatically. Serverless performs well during the data processing because of its scalability. In addition, testing/staging environments can be deployed cheaply due to not paying for a fixed amount of resource.

Serverless seems to be fit with event driven computing, stateless and short running and is not good for number crunching, stateful and long running which is mentioned in the paper [**5**] published in 2017. However, paper "The Rise of Serverless Computing" discuss about serverless applying in scientific computing. It is available to run functions with only paying for what is used, which makes users more convenient and focus on their experiments. Thus, serverless has gained popularity in compute intensive applications.

PyWren is a well-known application with map-reduce style framework for highly parallel analytics workloads[**2**][**3**][**5**]. It utilizes serverless framework to avoid the significant development and management overhead of running MapReduce jobs.[**2**] By using AWS S3 for storage caching, Barga mentioned PyWren 600 concurrent functions can reach 25 TFLOPS performance, 60 GB/sec reading and 50 GB/sec writing speed to S3. PyWren represents a type of use cases for highly parallel analytics workloads. If the workloads can be easily divided into small parts, serverless computing performs well on such job otherwise users should consider other technology such as high performance computing.[**2**]

The area where serverless has **advantage**:	The area where serverless has **disadvantage**:
(Usually short-running, stateless, event-driven)	*(Usually long-running, stateful, number crunching)*
Web appications	Data bases
Backends	Deep Learning Training
Chatbots	Heavy-Duty Stream Analytics
IT Automation	Spark/Hadoop Analytics
Data processing(Portionnal)	Numerical Simulation
Servive intergration	Video Streaming
IoT	computing workloads which is hard to divided

Figure 4: The areas of Serverless Computing having advantage or disadvantage

In recent years, as an emerging technology, serverless computing has been applied in more and more aspects like parallel analysis, API Composition[**2**], Multi-Tenant Cloud Services[**2**], etc. *Figure 4* summarizes some of aspects where serverless computing has advantage or limitation. Not hard to imagine that serverless computing will benefit other areas with the improvement of technology.

2.6 Future of Serverless Computing

FaaS has so many advantages, nevertheless FaaS is not widely used. This is mainly because its capability, like large data/function scaling problem, and limitations like time limitation, memory limitation. With the technology development, shortage will be eliminated. FaaS will be applied to general purpose computing. Let's see what change may happen in future for FaaS.

Long time service may be available in future[**5**], the provider may provide a service level agreement (SLA) in the future. But this brings challenge to providers, long time tasks bring more difficulty in cost-effective SLA's and give the provider less flexibility in scheduling and Legacy code is available for FaaS[**1**]. Nowadays, there are a lot of existing code and is using in commercial environment. If it is available to auto-decompose existing codes to FaaS version, or very few code modification by developer, FaaS will be widely deployed in future.

High performance and parallel programming related application may be applied[5], a programmer called Barga started his challenge to apply the MapReduce on FaaS model and interactive scientific

notebook will emerge to help people on development. Moreover, It will be bring to batch processing to FaaS to reach the performance on supercomputers.

Cooperate with Edge Computing[2][5], in big data ages privacy problem, big data transport time, varies demands of users made edge computing blooming and it has a very strong link with FaaS. The edge device send a request (trigger) to FaaS, the cloud send back a run-able container (workload) to the edge, then the edge deploys and solves related demand.

3 Evaluation on Self-Deployed FaaS Platforms

3.1 Apache OpenWhisk vs OpenFass vs Knative Intro

Apache OpenWhisk is an open-source platform for serverless computing with a large amount of underlying components. It supports many languages, such as NodeJS, Go, Java, Scala, PHP, Python, Ruby and Swift, as well as recent additions for Ballerina, .NET and Rust. Moreover, it can leverage CouchDB, Kafka, Nginx, Redis and Zookeeper, which brings both advantages and disadvantages. Developers can focus on scalability and resiliency while they may meet difficulties to master these tools. In addition, its security can be improved.[**15**][**18**]

OpenFaaS is a free and independent project using the Serverless Framework, which includes the following components: API Gateway, Function Watchdog and the container orchestrators Kubernetes, Docker Swarm, Prometheus and Docker (as shown in *Figure 3*). Developers can deploy event-driven functions and microservices to Kubernetes easily using OpenFaaS. In addition, functions can be written in any language for both Windows and Linux and packed in Docker or OCI image format. Moreover, it can run on hardware or public or private cloud. However, OpenFaaS has weaknesses. Firstly, the starting time may be long when using some programming languages. Secondly, the provider decides the start time of the container. In addition, containers cannot store code in memory for a long time.[**15**][**19**]

Knative is an open-source project based on the Kubernetes platform to deploy and manage modern serverless workloads whose architecture consists of the Building, Eventing and Serving components (as shown in Figure). Among the upsides of Knative are it can start the service in seconds and it supports common patterns such as GitOps, DockerOps and ManualOps. Secondly, Knative and other common tools and frameworks such as Django, Ruby on Rails, Spring, etc. can be used together when developing. Moreover, it is available for developers to create applications internally, such as in the cloud or in a third-party data centre. However, one of the drawbacks of Knative is the demand to independently manage container infrastructure.[**15**][**20**]

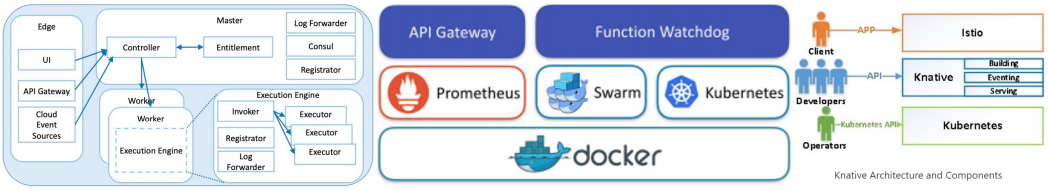


Figure 5: Apache OpenWhisk(Left) OpenFaaS(Middle) Knative(Right)

Now we have a good overview on three popular self-deployed frameworks, but how do developers choose the most appropriate platform for their apps? What is the computation characteristic so that we can know their own advantages and disadvantages? From computation aspects, [**7**][**9**] gave us a hint that we can test platforms from CPU Calculation Speed, I/O Speed, Memory Capability, Execution Time on Elasticity, Response Time for Dynamic Workload, Service Start Delay to know their performance. From the applicability angle, we can use big data streams[**8**], machine learning[**10**] regard as test cases. Besides, how hard is the platform to be deployed, maintain, related documentation quality and social support of the platform are also needed to be evaluated for our developer readers.

3.2 Experimental Setup

We used a recent version of Docker (CE 0.18.0) and Kubernetes (Client v1.18.1, server v1.15.11). AWS EC2 EKS cluster info: 3 nodes, each of m5.large type (i.e., each with 2 vCPUs, 8 GiB memory).

Item	OpenWhisk	OpenFass	Knative
Support laguage	NodeJS, Go, Java, Scala, PHP, Python, Ruby, and Swift as well as recent additions for Ballerina, .NET and Rust	Go, Java, Python, C#, Ruby - Express.js, Django, ASP.NET Core, even binaries like ffmpeg, ImageMagick	C#, Go, Java (Spark), Java (Spring), Kotlin, Node.js, PHP, Python, Ruby, Scala, Shell
Popularity (Github Stars)	4.7k	17.7k	2.6k
Contributors	860	147	174
Corporate Backer	IBM(Apache Foundation)	VMWare	NA
Started date	Feb 2016	Dec 2016	July 2018
Container	Docker	Docker	Docker
Serverless?	Yes	Yes (WIP)	Yes
Underlying technology	CouchDB, Kafka, Nginx, Redis, Zookeeper	Alertmanager / Prometheus, Nats	NA
Worked out of the box?	Yes	Yes	Yes
Quality of documentation	Good	Good	Good
Slack channel availble?	Yes	Yes	Yes

Figure 6: Comparison among Apache OpenWhisk vs. OpenFaaS vs. Knative

The serverless technologies CLIs, Docker CLI and Kubectl were installed on a client machine with 15,6 GB memory, Intel R CoreTM i7-4702MQ CPU @ 2.20GHz 8 Os-Type: 64-bit Disk: 128,5 GB. Because of the poor documentation of OpenWhisk, we were not able to deploy/install the OpenWhisk application on the AWS Kubernetes cluster. Also, for local deployment, some limitations (mentioned below) restricted us to do scalability and overhead tests for OpenWhisk, and at the end, we did the speed tests locally on a single node.

We used the AWS Kubernetes cluster and installed Knative and OpenFaaS. We created functions and deployed the function in their respective namespaces.

Pre-requisites for Knative :
1. eksctl CLI to connect to an AWS EKS cluster
2.AWS CLI to connect AWS
3.Configure AWS to connect with an AWS EKS cluster
4.Kubernetes CLI (i.e., kubectl)
5.Helm CLI - version 3

Pre-requisites for OpenFass:
1.eksctl CLI
2.AWS CLI
3.Configure AWS to connect with an AWS EKS cluster
4.Kubernetes CLI (i.e., kubectl)
5.Helm CLI - version 3
6.OpenFaaS CLI

3.3 Experimental outcomes

3.3.1 Apache Open Whisk Speed

With a limitation of 60 requests per minute, according to[**openwhisk-Docs**]], we need to test OpenWhisk by blocking and non-blocking requests.

Blocking invocations mean the platform won't send the HTTP response until the action finishes. This allows it to include the action result in the response. Blocking invocations are used when you want to invoke action and wait for the result.
Non-blocking invocations return as soon as the platform processes the invocation request. This is before the action has finished executing. HTTP responses from non-blocking invocations only include activation identifiers since the action result is not available (default behaviour).

As shown in the graph, non-blocking invocations are much faster than blocking invocations. The difference is 3.411 s because there is no waiting for the action to be finished.

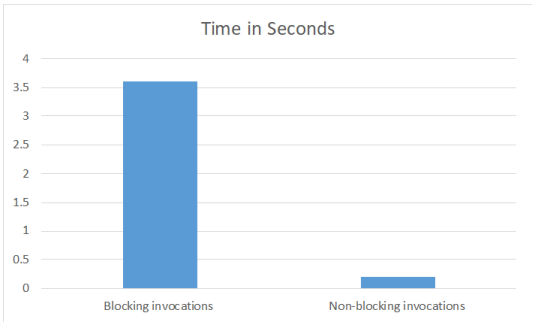


Figure 7: Apche open Whisk Speed Test

3.3.2 Speed,Scalability and Overhead for knative/openfaas

To test Knative and OpenFaaS, we deployed 3 sample Hello World applications written in Go programming language.

1. App 1 with 1 replica - invoked 100 times using 1 session 2 with 5 replicas - invoked using 3 concurrent sessions, 100 requests each
2. App 3 with 10 replicas - invoked using 5 concurrent sessions, 100 requests each

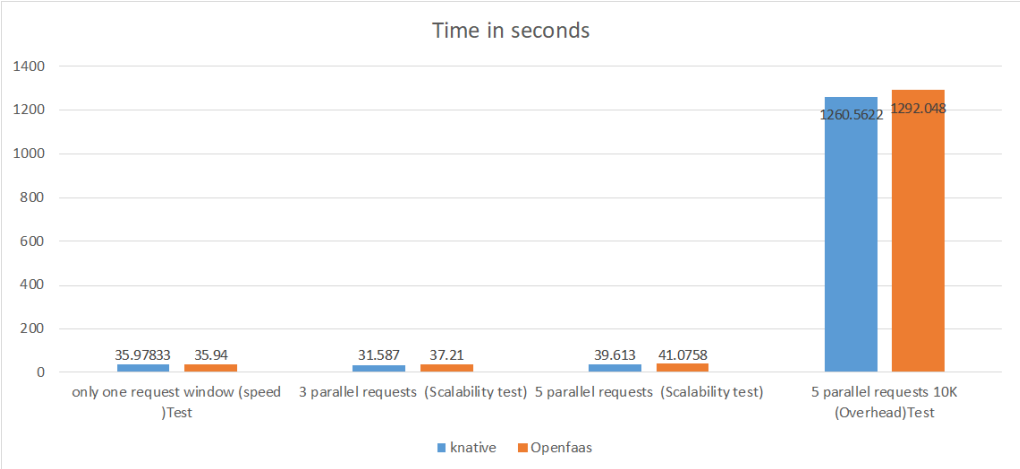


Figure 8: Speed, Scalability and Overhead (Knative vs OpenFaaS)

As shown in the graph, OpenFaaS is slightly faster than Knative by 0.03833 s.

Knative is more scalable than OpenFaaS.

OpenFaaS has more overhead requests than Knative.

3.3.3 Usability/Challenges

OpenWhisk: Documents on deploying OpenWhisk on the AWS Kubernetes cluster are rare. The instructions provided on the official document don't work as written or expected. They need to be improved, and the technology needs to be mature.

This was not the case for Knative and OpenFaaS. Both are well-documented, and the community is active.
Despite the growth of open-source serverless technologies, we believe Apache OpenWhisk is not

mature and very efficient in the production, so a single node of OpenWhisk may not be realistic in the production environment.

Knative and OpenFaaS are better, especially with the smooth integration with Kubernetes, which will make the platform scalable in creating, deploying, and managing serverless workloads.

Although we conclude from the experiments that Knative is better in scalability and overhead, OpenFaaS is an excellent platform for data scientist developers because it has built-in models for machine learning to analyse pictures, data, etc. Aside from its simple configuration and maintenance, OpenFaaS is more straightforward and allows a developer to install an entire component, while in Knative, the developer needs to install the components separately.

4 Conclusion

From papers we can see that serverless computing and not a really new technology in industry but a new trend with many new attributes, and its best implementation is FaaS. Serverless is a technology has three main attribute: high scalability, low cost (pay-as-user-demand), don't need expertise knowledge on server, and these advantages are changing the way that developers conduct services. Right now it is widely used on web applications, IoT and IT automation, however due it current limitation it still cannot applied to all kinds of applications, like stateless application, legacy application (exist code) and large scale application. We believe with the development of SC, limitation will be solved and more application will be applied.

As performance experiments on three opensource serverless frameworks, we found that the open whisk is not mature because of poor documentation for developers. Openfass and Knative are compatible with Kubernetes's high scalability attribute and have high performance and are best choices for developers.

References

[1] Baldini, I. et al. (2017) Serverless Computing: Current Trends and Open Problems.

[2] Castro, P. et al. (2019) The rise of serverless computing. Communications of the ACM. [Online] 62 (12), 44–54.

[3] Lee, H. et al. (2018) 'Evaluation of Production Serverless Computing Environments', in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). [Online]. July 2018 IEEE. pp. 442–450.

[4] Malawski, M. et al. (2017) Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions. Future Generation Computer Systems. [Online]

[5] Fox, G. C. et al. (2017) Status of Serverless Computing and Function-as-a-Service(FaaS) in Industry and Research. [Online]

[6] Leitner, P. et al. (2019) A mixed-method empirical study of Function-as-a-Service software development in industrial practice. The Journal of Systems Software. [Online] 149340–359.

[7] Klimovic, A. et al. (2018). Understanding Ephemeral Storage for Serverless Analytics. Stanford University. [Online]

[8] Kiran, M. et al. (2015) Lambda architecture for cost-effective batch and speed big data processing. [online]. Available from: https://escholarship.org/uc/item/0t36p3hn.

[9] Hendrickson, S. et al. (2016). Serverless Computation with OpenLambda (Unpublished doctoral dissertation). University of Wisconsin—Madison. [Online]

[10] Lin, W.-T. et al. (2018) 'Tracking Causal Order in AWS Lambda Applications', in 2018 IEEE International Conference on Cloud Engineering (IC2E). [Online]. April 2018 IEEE. pp. 50–60.

[11] Ye, W. et al. (2003) 'Distributed network file storage for a serverless (P2P) network', in 11th IEEE International Conference on Networks, 2003. ICON2003. [Online]. 2003 IEEE. pp. 343–347.

[12] Bryan, D. . et al. (2005) 'SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System', in First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA'05). [Online]. 2005 IEEE. pp. 42–49.

[13] Johnston, P. (2017, September 08). A simple definition of "Serverless". Retrieved May 23, 2020, from http://bit.ly/2G3Hp1R

[14] Brazeal, F. (2018, September 25). Serverless is eating the stack and people are freaking out - as they should be. Retrieved May 23, 2020, from http://bit.ly/2xzNEWB

[15] Serverless Open-Source Frameworks: OpenFaaS, Knative, amp; More. (2020, February 27). Retrieved May 28, 2020, from https://epsagon.com/blog/serverless-open-source-frameworks-openfaas-knative-more/

[16] OpenFaaS Ltd. (2020, January 14). OpenFaaS 3rd Birthday Celebrations. Retrieved May 28, 2020, from https://www.openfaas.com/blog/birthday-teamserverless/

[17] IT Automation. (n.d.). Retrieved May 28, 2020, from https://www.vmware.com/topics/glossary/content/it-automation

[18] A Comparison of Serverless Frameworks for Kubernetes: OpenFaas, OpenWhisk, Fission, Kubeless and more. (2018, September 01). Retrieved May 28, 2020, from https://winderresearch.com/a-comparison-of-serverless-frameworks-for-kubernetes-openfaas-openwhisk-fission-kubeless-and-more/

[19] Project, O. (n.d.). Introduction. Retrieved May 28, 2020, from https://docs.openfaas.com/

[20] Knative. (2020, May 12). Retrieved June 01, 2020, from https://knative.dev/

Contribution

4.1 Preparation Stage

Tianyang Lu did search for general intro, discussion, evaluation paper on Serverless and Faas (About 15 papers)

Hatim did search for general intro, discussion on Serverless and Faas (About 5 papers), did research and experiment on Apache OpenWhisk , OpenFass and Knative. Also find overleaf template for group work.

Kai over-viewed all papers & conducted works, and made the literature overview skeleton. Then set up a meeting and distribute works with group members' advantage. Also added 5 supplementary paper and related web pages for Apache OpenWhisk , OpenFass and Knative.

4.2 Writing Stage

Kai is responsible for Abstract, section 1 introduction, section 2.1 Definition and Architecture 2.2 Characteristics 2.3 FaaS Compare to other Cloud Structures 2.4 Status in industry and 2.6 Future of serverless-computing.

Tianyang is responsible for section 2.5 Use cases, and rewrited section 3.1 Apache OpenWhisk vs OpenFass vs Knative Intro. Kai add instruction in section 3.1 for section 3.3.

Hatim was responsible for all section 3, but 3.1 but Tianyang Updated This Section and rewrite some of it. Hatim now is responsible for section 3.2 experimental setup 3.3 Experimental outcomes.

Kai and Hatim wrote the conclusion and Kai is responsible for restructure.

4.3 Review Stage

All guys did review on Abstract and Introduction.

Kai did review on Tianyang's part, and gave suggestion that this part is better to add illusttration with SC's characteristics.

Tianyang did review on Hatim's section 3.1, and found there are a lot of redundancy information. So she had to rewrote it.

Hatim's part section 3.2, 3.3 is not being reviewed by others, because no other group members knew how did he do the evaluation and why. He conducted review on his own part.

4.4 Final Stage

Kai conducted structure overview through out the paper to make the paper read smoothly.

Tianyang updated all figures and citations.

The status of Containers and Unikernels from a cloud perspective

Jackson, Alappatt Warrunny
12842753

Giannakopoulos, Athanasios (Thanos)
12747300

Abstract

Over the last few years, virtualisation and containerisation gained huge momentum within the IT community by incorporating these in providing cloud-native solutions which they offered. Although containers overcame many constraints posed by monolithic inflexible and unchanging practices of packaging, sharing and deploying applications, it is imperative and critical to understand the limitations or weaknesses they have. The main constraint of the containers (e.g. Docker) is related to the fact that they are not truly sandboxed as they share the host OS kernel. This weak separation results in the situation where the host OS kernel creates a virtualized userland for each container everytime with potential security threats on the environment. The aim of this paper is to make a thorough analysis of the evolutions of recent technologies and solutions which target to eliminate this isolation weakness. Moreover, this paper will discuss how well Unikernals solves this problem and make a comparison between the two in terms of their capabilities.

1 Introduction

Containerization is the virtualisation technique which allows to package applications and run them in isolated environments. Due to this offered feature that containers provide, they have become really popular in the cloud application management. This technology offer flexibility and convenience to easily deploy applications in the cloud. Development teams working in an agile environment especially DevOps, can work independently and without interfering to the others using development pipelines easily adhering to the DevOps principles. Docker technology was the first to try to popularize Linux containers. Client server, registries, images and containers are the main components of the Docker technology which will be described with more details in the paper later on. Docker started to gain its momentum in 2013 as the chosen open source approach for software production had already been widely accepted by the Information technology community. Other than open source approach there are other reasons which made Docker the dominant technology for many years. Among the main benefits that containers offered to the community were the lightweight virtualization and standardization. However, containers brought also some limitations/weaknesses as well. The main limitation was potential security threat which is based on the fact that they share the host OS kernel among multiple containers exposing it possible data privacy breaches [2].

In this paper we try to assess whether Unikernals is a right answer to this limitation of Container. Unikernals were born recently which provides a way to quickly build and deploy relatively small applications on the VMs that do not require functional and operations functions that would typically offered by containers. It provides good degree of isolation as every Unikernal is packaged with the absolute required OS kernel components together with the application to be deployed. In the recent years it has been attempted to mature Unikernals from a beta version to productonise supporting new and existing applications.

2 Containers

2.1 An introduction to Containers

Containerisation technology with its unique attribute of being capable of virtualising applications in a lightweight, flexible and easily scalable manner, has led to its wide adaptation in cloud application management. More specifically, containers are an operating system virtualization technology used to package applications and run in the target platform. This technology allows to run the applications in isolated environments along with their dependencies. They provide a lightweight method of sharing, packaging and deploying applications in a standardized way across many different types of infrastructure [9]. In other words, containers offer a way in which applications can be packaged totally independent and separated from the environment in which they actually run. This decoupling makes the deployment easier and more consistent for container-based applications, regardless of whether the target environment is the cloud, a server-based ERP software or even a personal laptop. This guarantees that all the processes running inside a container will not have any interference with any processes outside it as depicted in Figure 1. As a result of this, every role in a production team should have separate concerns and responsibilities. Developers are able to focus on their application logic, dependencies and fixing any bugs, while IT operations teams can focus on deployment and of course management with limited knowledge of the application details. That means that any configurations and new versions of the software can be tested and handled without causing any inconvenience to the final users [6].

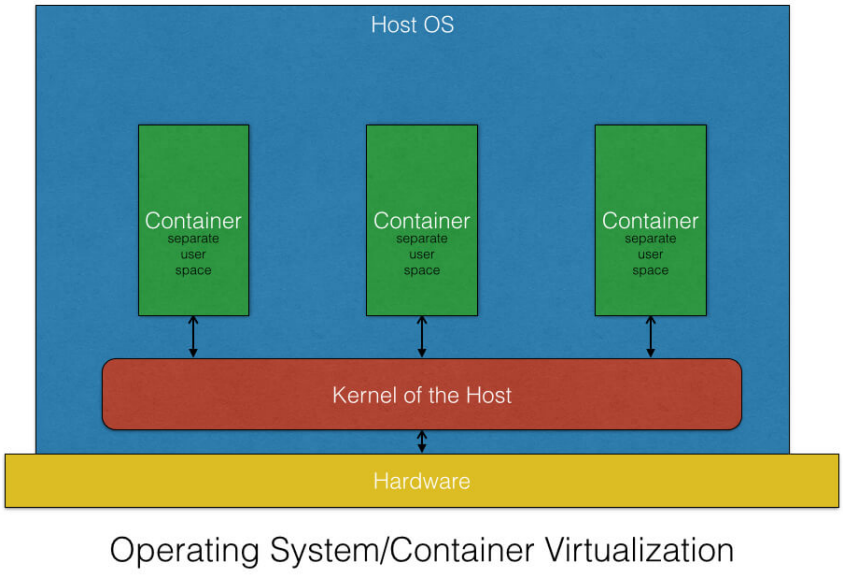


Figure 1: Containers

2.2 Why Containers

In this section we will focus on relevance of containers over Virtual machines. It is important to understand how the features offered by these differ. The primary difference is in their virtualization level. Virtual machines offer virtualization in terms of the hardware stack while containers virtualize at the operating system level. Multiple containers can run at the OS kernel directly. As a result, there is a comparative advantage for containers as they become far more lightweight. Containers share the same OS kernel and use only a fraction of the memory while booting up instead of booting an entire

OS which makes them to start up faster. [6].On the other hand, comparatively heavier virtual machines can cause several issues if there is a need to apply changes to the application's environment. For instance, this could happen when the changed software is promoted to a higher level up until production where two environments could be exactly the same. This lack of flexibility in porting is a strong limitation which can be resolved using containers. Portability can be really simple and easy procedure using containers in the cloud. Applications requiring a migration from one cloud environment to another using containers requires far less effort. A risk of failure is much lower considering container's flexibility and integrity of code together with its dependencies regardless of external factors.

2.3 Container technologies

As we have discussed earlier, containers offer an efficient way to run applications along with their supporting dependencies in an isolated environment. Docker is the open source platform which facilitates the independent running of these application in a simple and easier way. The first technology to popularise Linux containers was Docker. The major components that makeup the Docker includes Docker Client and Server, Docker images, Docker Registries and Docker containers. These internal components are explained in more detail in the following sections. [1]

Docker Client and Server: Figure 2 below represents Docker as a Client/Server based application. The starting point is a new request from docker client towards the docker server which then process it accordingly [15]. Docker is responsible for the shipment of the complete RESTful(Representational state transfer) API and the command line client binary. It is worth to note that there are two possible scenarios in which the Docker server and client can reside. The first scenario is that both Docker Daemon/server run on the same machine. In contrast, it is also possible that the local client can be connected with a remote server/daemon which is running on a different machine.

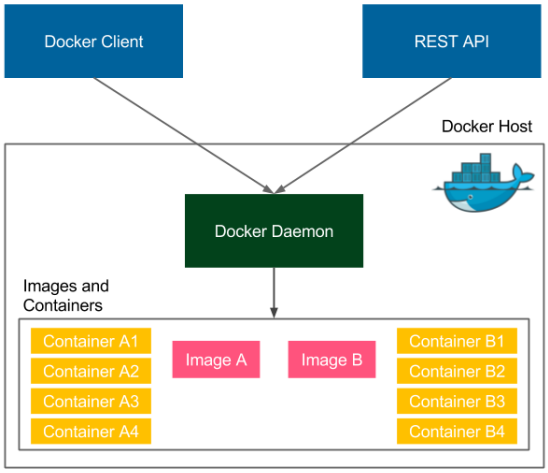


Figure 2: Docker Architecture

Docker Images: Docker images are files that are used to execute code in a docker container. The process of building a new image is called "committing a change". There are two methods to build an image. The first one is the process to build a new image from scratch. It is usually built using a read-only template. The second method is to create the image using a docker file. The docker file which contains a list of instructions is executed with "Docker build" command from the bash terminal. This process will follow all the instructions given in the docker file and builds an image. This is an automated way of building an image than using the first method.

Docker Registries: A docker registry is a storage to hold docker images. It also acts as the distribution system for docker images. These could be same images with different versions which are identified using tags. As like a source code repository, the docker registries can be viewed as the source where images can be promoted or demoted. There are two types of registries, public and pri-

vate. By default, the docker engine interacts with Docker hub which is the public registry instance. From this docker hub, the images can be pushed or pulled without creating them from the scratch.

Docker Containers: A docker container is an abstraction of application layer which packages code and its dependencies together so that the application can run in isolation and as a whole. Several containers can run on the same machine at the same time and can share the OS kernel with others. However every containers runs their process in isolation from others.

2.4 Popularity

We have already discussed why containers, in general, have proven so appealing to companies large and small over the past several years. Let us now have a detailed look at Docker.

Docker was open source from the start. In 2013, the IT community was beginning to adopt the open source as the standard mode of software production and sharing. This was definitely one of the most important factors that made the Docker so appealing in the technology market. [11].

Docker got introduced at the right time. By the time Docker was released as a technology, virtual machines were finally becoming an outdated technology. IT industry was looking for a new, different and more simple way of deploying applications and Docker was available at the right time. [11].

Docker enabled with the DevOps way of working. DevOps, which became more of a norm in the early 2010s, emphasizes agility, flexibility and scalability in software delivery. Docker containers happen to provide an excellent building block for creating software delivery pipelines and deploying applications according to DevOps prescripts [11].

Docker coincided with the Agile revolution. Agile approach particularly CI/CD (Continuous Integration/Continuous Deployment) in software delivery was widely accepted as a mechanism to develop and deploy IT components. The framework brought flexibility and scalability compared to the traditional waterfall methodology. Docker could very well support DevOps approach for developing software pipelines according to these new standards.[11]

2.5 Benefits and disadvantages

Now that we have discussed about containers and related technologies, let us also analyse the benefits and limitations/disadvantages of containerization. [7].

2.5.1 Benefits

As it is already mentioned, containers are a streamlined way to build, test, deploy, and redeploy applications on multiple environments. In addition to that, we discussed about containers technologies and the popularity of containers. In this section we will try to give an overview of the benefits they offer. Benefits of containers, amongst others, include the following: [18]

Lightweight Virtualization: Containers are lightweight compared to Virtual machines. Virtual machines virtualizes all the hardware resources using a totally independent operating system while containers share host OS kernel and run as separate isolated processes within the host.

Containers are faster in startup and stopping and they do not need many system resources. Container acts as a total independent environment even if it shares only a subset of host's resources.

Usually the container images are smaller in size that allows the workflow which pulls down latest image at runtime to perform efficiently without much delays. This feature is very significant for many fault tolerant, self-healing distributed systems.

Isolation: Since Containers use Linux kernel features, it helps them to isolate itself from other containers. As a result there is no interference among them. In addition, there is also no conflict in dependencies and libraries that each container use as they are maintained in separate file system. Hence it is safe that applications within the container do not conflict with those on others. This helps the administrators to focus on the mapping between container's networking and host networks based on their requirements.

Standardization: One of the most appealing benefits of containers is that they offer a simple way of packaging and deploying software. In the docker technology section, we discussed that container

images allow developers to run applications along with their supporting dependencies into a single isolated unit. That is to say, developers are able to install and use all the necessary libraries inside the containers, without any risk of interfering with corresponding host system libraries. When a container image is being created, supporting dependencies are labelled to a specific version each time. Next to that, developers need not be aware of the configuration details of target environment where the container will be running. Operationally wise, administrators can work easier and focus on maintaining generic hosts that operate as container platforms. In other words, there is a clear logical separation between the application and the infrastructure platform which requires less effort from developers and administrators.

2.5.2 Disadvantages

Although containers has some advantages over a virtual machine, they have some limitations as well. [3]

Sharing the OS host: The major weakness of containers is that they share the same kernel of host OS and as a result are less isolated than real VMs. This kernel dependency can cause serious issues as a glitch in the kernel could affect every containers running on it. On top of that any security issue could be catastrophic because multiple containers running on the same host may belong to other tenants.

Complexity: It is possible that multiple containers can run on the same host. When the number of containers increases, a corresponding increase in the complexity factor is to be considered as well. Developers could face a challenging task to maintain the different containers in the production without a proper tool for management.

3 Unikernels

3.1 An introduction to Unikernels

In its most simplistic term, a Unikernel looks to be an 'improved container' but with better security and performance, it brings. It is a collection of Linux kernels which allows to package an application in such a way that it makes it more difficult to interfere with others. It may be described that Unikernels are applications running without an operating system. However, it is technically not true because Unikernels has operating system but its not like a general purpose operating system such as Linux. Typically a general purpose operating systems has several processes that starts running when the system bootsup even when no applications are running on it. However, Unikernels has only one process which is the application that is designed to run on it. Along with this, it uses absolutely minimal software stack in the form of required operating system components. This minimalist requirements means reduced attack surface for any kind of intrusion. This feature, unlike containers, makes Unikernels most secure and runs with high performance.

3.2 Evolution of Unikernels

Developer community is always in the quest to modernise the infrastructure to deploy their applications in the cloud by fully utilising the virtualisation, light-weight, fully secured and highly performing environment with minimal memory foot-print. Unikernal is a natural evolution enhancing the attributes that are offered by virtual machines and furthered by Containers. Unikernal is the latest cloud-era solution which links the application with a library of various operating system components that are required to run the application. These components including memory manager, scheduler, network stack and device drivers which are integrated into a single address space generating a binary image that is bootable directly on a virtual hardware [13]. This feature to include only the required operating system components packaged together with the intended application, brings the improved performance and the required security since the exposed surface is very limited for any potential attack. In the recent years there had been a spike in introduction of various Unikernels which target to utilise cloud and internet workloads. Network performance became key for such cloud hosted systems and Unikernel's offering of simplified IO paths and removal of domain crossing helped to improve the latency and throughput of such network intense solutions. It is proven that Memcashed running on a Unikernal TCP/IP stack gave a throughput improvement of over 200 percentage when compared to that of a Linux [16]. Also it has been shown that Unikernals deployed

within a MicroVM performed significantly higher in terms of requried boot time, when compared to containers [12]. Similarly a micropython unikernal had an image size of 1MB and required 8MB at execution [14]. Despite the high security feature and performance benefits offered by the Unikernel, it is yet to be widely adopted outside the domain of research and experiment platforms. It is perceived by the developer community to have much more effort required to deploy an application to a runtime with only a partial support to legacy software interfaces is provided by Unikernels.

3.3 Unikernel Architecture

In this section we will focus on isolation technology which is the salient feature of Unikernels and how it is achieved. In a multi-tenant cloud environment mutually mistrusting users share the same physical resources. While isolation can be established from a hardware perspective (e.g page tables), here we will discuss the potential security leakage which could exploited via the software interfaces. The threat could arise from the malicious tenant that could break out from its isolation seriously interfering with others. The level of this security compromise depends on the attack surface that are exposed. While Unikernals uses bare minimum OS components just sufficient to run the targeted application, this threat is far less compared with containers, for instance.

Most of the operating systems used today in the Virtual machines are identical to the traditional general-purpose operating system that is also used in non-virtualised environments. It is common practice these days that virtual machines are configured to run a specific application intended for its own use such as a web server or database together with the required code. In this context most of the package and services offered by a general purpose operating system is not used. Moreover such unused components increases the risk profile from a security point of view for the infrastructure on which various tenants are running. Figure 3 below shows the high level architecture representation of Traditional OS structure vs Unikernel.

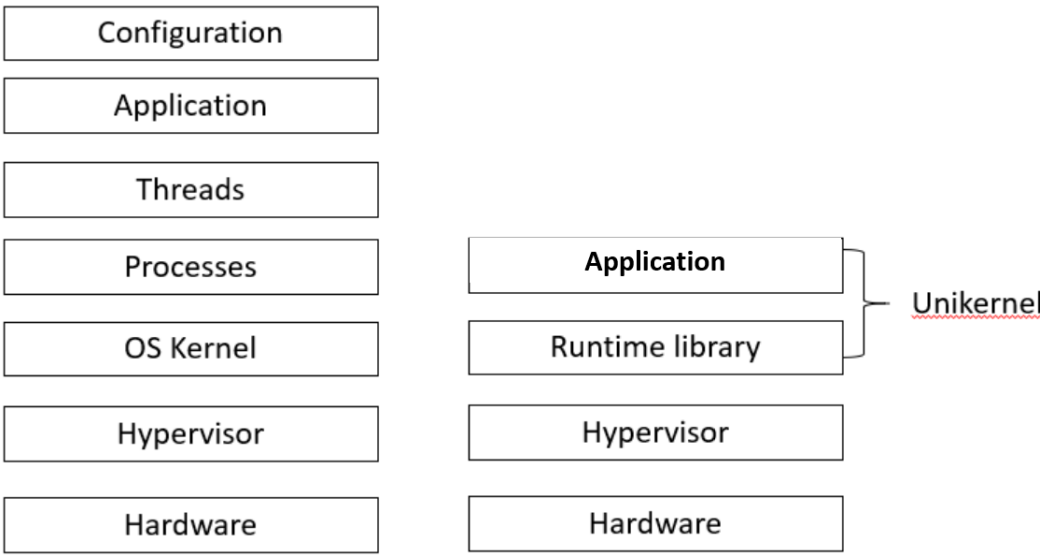


Figure 3: Traditional OS structure vs Unikernels

The figure 4 below shows the Unikernel Ecosystem how it sits within a typical development environment [17]

The creation of new Unikernel follow one of the two approaches: A clean state approach where the kernel is created from the scratch, or a strip down approach where the kernel codebase is stripped off from the unnecessary components making it fit for the application that it has to run.

Few solutions to construct unikernels with different level of maturity are described below.

MirageOS is one of the most famous Unikernel projects which uses clean slate approach to construct Unikernels. MirageOS can be used to construct Unikernels for various cloud computing and mobile platforms offering a library operating system. It is typically used in an environment which hosts

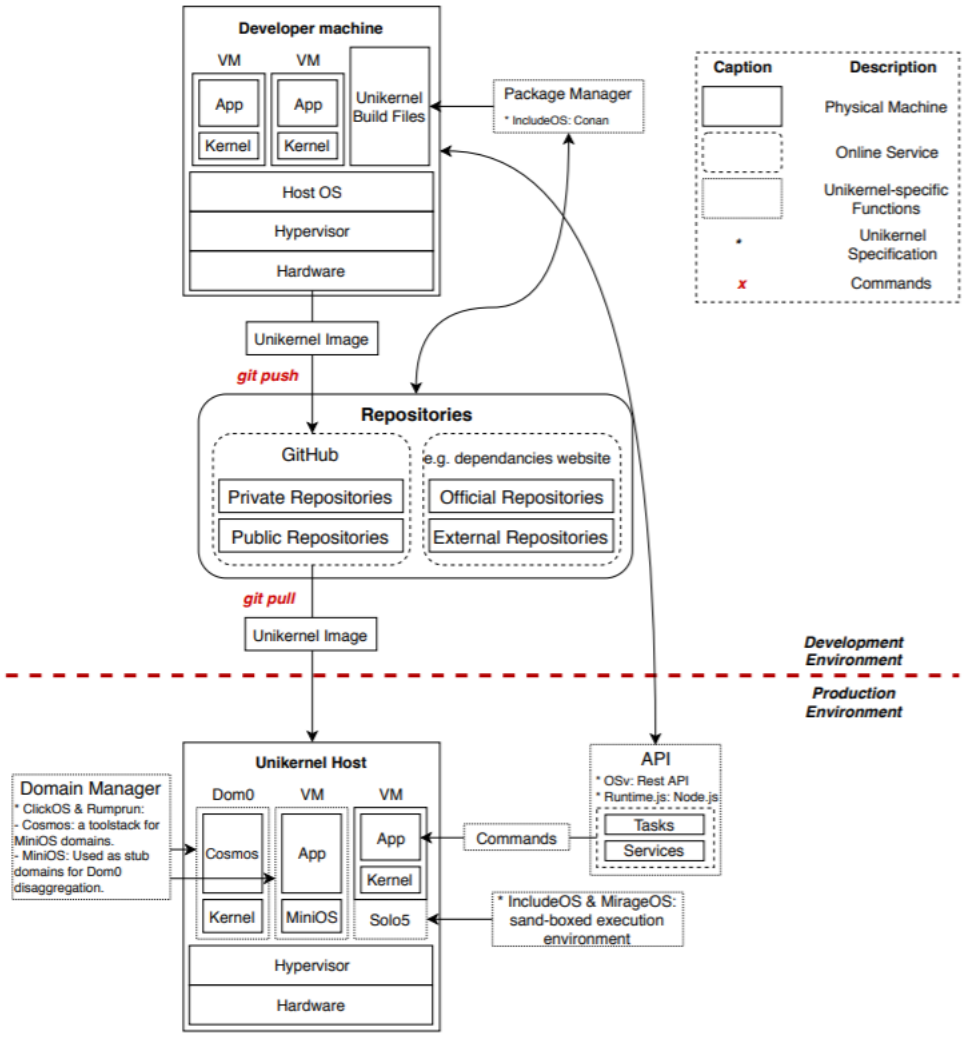


Figure 4: Typical Unikernel Ecosystem in a development environment

high-performance and secured network applications. In the first stage, code can be developed on a normal operating system(Linux, MacOS X), post which the standalone Unikernel can be integrated with the developed code. In execution mode, a hypervisor is responsible to run this specialised Unikernel. This framework allows all the hosted services to run more efficiently and secured. All these state transitions contains different events which gives this framework an event-driven feel eliminating the need to support proactive failures [8].

ClickOS is another example of a Unikernel project in an open source virtualization platform ideal for high-performance applications. Based on their performance, ClickOS Virtual Machines are super small (5MBs), have fast boot time(less than 20ms) and little delay(45ms). On top of that, more than 100 clickOS VMs can be run at the same time on a server [5].

Another well-known Unikernel project is **LING**. This Unikernel project follows Erlang/OTP protocol and understands .beam files. The code can be created in Erlang and deployed as LING Unikernels. An advantage of LING is that it removes the majority of vector files and uses only three external libraries and no OpenSSL[5].

3.4 Benefits and limitations of Unikernals

Having described what Unikernels are, its evolution, architecture and different Unikernels projects, an overview of its advantages and disadvantages are described in this section.

3.4.1 Benefits

Security: The offering that Unikernels provide on security is the most important advantage among others. Cyber threat is a major concern for organisations especially hosting their service in the cloud. These could be intentional or unintentional but both provide a major threat to the reputation of the organisation in terms of data security. The IT unit of most organisations are struggling to avoid any potential lapse in security leading to their never ending quest for a security efficient solution to run the applications on the cloud. Unikernels having its minimalist approach reduced this security risk by having a smaller attack surface which it exposes. An example, among others, will help us to understand better the strength of isolation which is provided by Unikernels. In case of a potential attack in the form of wrongdoing via accessing password file, shell and utilities programs, it is simply impossible because they just do not exist [10].

Size and speed: Based on the ClickOS projects that we discussed in the Architecture section, it is obvious that Unikernels are really fast and small in size. A typical working domain name server of MirageOS can be compiled into just 449 KB. Based on such performance statistics, it is obvious that size and speed are really strong benefits of Unikernels [10]. However, it is to be noted that on heavy workloads Unikernals tend to provide same speed as containers.[4]

3.4.2 Limitations

Impact on software development: It is expected that Unikernel is to support only a single process or thread to take advantage of its offering[3]. This limitation poses difficulty in its adaptation for legacy applications where they are already designed to run multiple processes and threads.

Unavailability of protection rings: Protection rings are used in traditional operating systems to provide increasing level of access. Since Unikernel packages OS components and applications together in the Ring 0, the flexibility of additional protection rings are not available [17]

No parallel processing: Unikernels are designed to encapsulate OS components and a single process. Hence anything that will require parallel processing will need to be broken down into several Unikernels.

Memory consumption: In terms of memory consumed, Unikernals seems to occupy high memory compared to containers. This roots from the reason that Unikernals has extra overhead on the kernels while Containers uses the kernel of host OS. [4]

4 Discussion and Conclusions

Container technology allows to run the applications in isolated environments along with their dependencies. However they depend on kernels of the host OS, which poses degradation in various aspects explained earlier. On the other hand, the concept behind Unikernels is to use the minimum software which an application needs. They have their own OS kernels providing better isolation and better overall performance.

Unikernel does better than Containers in terms of speed and response time, while containers seems to have advantage over Unikernels when memory foot print is considered. In terms of Security threat aspect, Unikernals are better placed than Containers with less threat surface exposure having only the required OS components packaged into the kernel together with the target application. However, Unikernels are yet to mature for industry wide usage although several initiatives are ongoing in the research areas. If the legacy applications can be split up or multiple instances can be used, or multi threading is not a requisite, then this limitation of Unikernals of not supporting multi-threading can be bypassed. While VMs provide good isolation but with heavy weight, Containers provide lighter-weight vitualisation. However, Unikernels although yet to fully mature, provides secured isolation, lighter-weight and comparable performance.

References

[1] Babak Bashari Rad et al. “An Introduction to Docker and Analysis of its Performance”. In: *IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.3, March 2017*. IJCSNS.

[2] Gundars Alksnis et al. “Containers for Virtualization: An Overview”. In: *Applied Computer Systems)*. De Gruyter. 2018, pp. 21–27.

[3] Alfred Bratterud, Andreas Happe, and Robert Anderson Keith Duncan. “Enhancing cloud security and privacy: the Unikernel solution”. In: *Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 19 February 2017-23 February 2017, Athens, Greece*. Curran Associates. 2017.

[4] Tom Goethals et al. “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)*. IEEE. 2018, pp. 1–8.

[5] http://unikernel.org/projects/. In: Unikernel.org.

[6] https://cloud.google.com/containers. In: Cloud.

[7] https://dzone.com/articles/container-technologies-overview. In: Dzone.

[8] https://mirage.io/. In: Mirage.

[9] https://rancher.com/blog/2019/an-introduction-to-containers. In: Rancher.

[10] https://techbeacon.com/enterprise-it/containers-20-why-unikernels-will-rock-cloud. In: Techbeacon.

[11] https://www.channelfutures.com/open-source/why-is-docker-so-popular-explaining-the-rise-of-containers-and-docker. In: Channelfutures.

[12] Ricardo Koller and Dan Williams. “Will serverless end the dominance of Linux in the cloud?” In: *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. 2017, pp. 169–173.

[13] Anil Madhavapeddy and David J Scott. “Unikernels: Rise of the virtual library operating system”. In: *Queue* 11.11 (2013), pp. 30–44.

[14] Filipe Manco et al. “My VM is Lighter (and Safer) than your Container”. In: *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 218–233.

[15] Claus Pahl et al. “Cloud container technologies: a state-of-the-art review”. In: *IEEE Transactions on Cloud Computing* (2017).

[16] Dan Schatzberg et al. “EbbRT: a framework for building per-application library operating systems”. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 671–688.

[17] Joshua Talbot et al. “A Security Perspective on Unikernels”. In: *arXiv preprint arXiv:1911.06260* (2019).

[18] Qi Zhang et al. “A comparative study of containers and virtual machines in big data environment”. In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE. 2018, pp. 178–185.

Contribution:
Athanasios (Thanos) Giannakopoulos : Containers, Review of Unikernels
Warrunny Alappatt Jackson: Unikernels, Review of Containers and preparation of presentation material

Edge Computing and Relation to Cloud Computing

Futong Han
Msc Computer Science
University of Amsterdam
12581135
hanfutong0804@gmail.com

Junyan Li
Msc Computer Science
University of Amsterdam
12566349
junyan.li@student.uva.nl

Binjie Zhou
Msc Computational Science
University of Amsterdam
12544752
zhoubinjie201909@163.com

June 1, 2020

Abstract

Edge computing is developed with the growth of Internet of Things devices. It is a decentralized computing architecture which aims to bring computation as close as to the users as possible, to provide the lowest possible latency. It makes use of IoT devices. Cloud computing, on the other hand, provides on-demand availability of computer system resources, and does not consider latency by default. The distributed nature of edge computing and its target at a large amount of unreliable devices bring new issues and challenges compared to distributed cloud computing. These two approaches focus on different goals, but they are not mutually exclusive. They need each other in many use cases. Business and IT companies fuse the strengths of them which work together and provide comprehensive solutions to IT infrastructure.

1 Introduction

Smart devices connected to the network first appeared in 1982. Carnegie Mellon University's improved Coke machine became the first network-connected machine. It can report its inventory and be able to detect whether the beverage is cold. The world around things connected to the Internet officially kicked off. *Internet of Things* (IoT) was first proposed by Professor Ashton in 1999 and then a large number of cloud computing and Internet of Things applications are gradually integrated into people's lives.
However, there are some problems such as slow response, privacy issues, broadband restrictions in cloud computing and in 2015, Garcia came up edge computing to help with these problems.[5] This report gives introduction of edge computing, cloud computing and discuss the relation between these methods.

2 Edge Computing

Edge computing is a kind of decentralized computing architecture. Online services, applications and data are moved from node of network to the network logic edge node for processing. The network edge is a functional entity related to the data source and the cloud computing center. Those things are together with edge computing platforms which integrate network, computing, storage, and application core capabilities to provide dynamic, efficient, and intelligent service computing for users.

Edge here means the network resources along the path between data source with online cloud center. There is a principle that the edge computing is a method of processing where the calculation happened closed to the source of data.

The edge node is much closer to terminal equipment compared to cloud computing because it can speed up the data processing and reduce response time. In this architecture, the generation of data are closer to the source of data, so they are easy for processing big data.

2.1 Reasons for Edge Computing

Why Edge Computing is necessary for the modern society? There are several reasons that edge computing has become one of the most important methods for computing service.

From cloud services

It has been proved that putting all the computing tasks on the cloud is an effective data processing method, because of the speed of computing power on the cloud exceeding the ability of the edge things. However, compared with the rapidly developing data processing speed, the network bandwidth has not kept pace. With the growth of edge data, the speed of data transmission has become the biggest problem of cloud computing model. Also, self-driving cars, which generate 1G data per second, need to process vehicles in real time to make correct real-time decisions. Due to this problem, the data from edge needs to be processed at edge to shorten the response time, process more efficiently, and ease the internet pressure.

From the Internet of Things

Nearly all the electronic devices is becoming one of the Internet of Things. These device plays the role as data producers and consumers, such as LEDs, traffic lights, even intelligent cooks. It can be inferred that things on the edge of the network will grow more and more in a short time. Therefore, the original data generated by these things are huge, and using cloud computing is inefficient to process all of this data. It means that data generated from Internet of Thongs will not transferred to cloud, but consumed at the edge of network.

The amount of data at the edge is too large using cloud computing, and it will result in huge unnecessary bandwidth waste and the use of computing resources. Then the need for privacy protection has also become a hidden danger of cloud computing in IoT. At last, most of the nodes in IoT are limited, and by offloading computing tasks to the edge can make all the system more efficient.

From data consumer to producer

Edge terminal devices always act as a data consumer. In edge computing era, things are not only data consumers now, at the same time they are also function as data producers. Since it is always private to collect physical data at the edge of network, privacy of users can be protected by processing data at the edge than uploading original data to the cloud. Also, things can not only request services at the edge and content from the cloud. They can also perform computing tasks from the cloud. With these tasks in the network, the edge itself has to be well designed in order to meet requirements in services, such as reliability, security, and privacy protection.

2.2 Summary

In edge computing, the purpose is to put the calculation near the data source and there are several advantages:

1. Edge application services significantly reduce the amount of data that must be moved, traffic, and distance that must be transmitted, thereby reduce transmission costs, shortening latency, and improve service quality.

2. Edge computing can use the same architecture and basic underlying computing technology as other clouds, and it can use the same cost curve.

3. Edge computing makes the data much more reliable, safety and privacy.

3 Cloud Computing

Cloud Computing is on-demand delivery of computer system resources over the Internet with pay-as-you-go pricing. The main types of Cloud Computing are as follow: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). It is a relatively new concept which was officially coined in 1996 within a Compaq internal document.[2] It mainly introduce the basic idea of *Internet Cloud*. It realizes the evolution of IT solutions can be caused by the more often use of applications in the Cloud services. Based on IEEE Xplore search result, there are 58,762 conferences, 7,232 journals and 376 books related to Cloud Computing from 1996 to 2020 (accessed 28/05/2020). And 8,264 conferences, 2,609 journals and 33 books are doing researches about Cloud Computing within 2019 - 2020 (accessed 28/05/2020). It can be seen from these information that Cloud Computing is a trendy topic and it is regarded as a valuable field of scientific researches.

3.1 Applications and Challenges of Cloud Computing

Cloud Computing benefits users by providing services such as data storage, computing power and the access to databases. This means storing and accessing data and programs are no longer only the work of PC's hard wire. It may sounds far from real life, but Google Drive as a popular storage application is indeed a cloud storage service. Another well known cloud service provider is Amazon. Amazon Web Services (AWS) is one of the most famous and powerful cloud service platforms. Here Amazon gives explicit benefits of Cloud Computing: agility, elasticity, cost-saving and globally deployment in minutes. [17] The customers of AWS range from single IT team to big corporations, and the reason behind it is the increasing complexity of tasks in the industry. [6] Cloud

Computing brings the development of IT into a new stage. It offers brand new opportunities for outsourcing data and thus benefits individuals, small business and healthcare systems. It provides the same leverage which was only hold by big corporations.

In the paper written by Mehdi Bahrami and Mukesh Singhal, they propose an idea of building a Service-Oriented cloud architecture. [1] In this way, the information can be transmitted without extra work on transformation. Here it focuses on eHealth systems and gives assessments of performance based on the Scalability, Customization, Independence of services and Standardization of service. However, it did not provides details of experiment setup, the plot derived from the performance of designed Cloud systems miss the details of how it counts the weight of all elements. And the algorithm used also neglect the selecting process. However, it still shows the potential of Cloud Computing in the healthcare field and gives useful opinion on how to reduce the impact of heterogeneity of systems to improve the efficiency.

Another article views Cloud Computing as a classification, business models, and research directions which is written by Christof Weinhardt, et al. [16] It present the Cloud Business Model Framework (CBMF) as a three-layer system, which are infrastructure in Cloud, platform and applications. It points out that more and more big corporations has realized the vital role of cloud services with increasing demand in modern industries. For example, Amazon, Google and IBM are all extending computing infrastructures and platforms as a core service. However, collected data is missing in the report and further information about pricing strategies are not presented. The highlights of this report is the Application Domains chapter. It emphasises the urgency of having a new business models, especially with respect to the licensing of software as well as an efficient and clear pricing strategy for complex cloud services.

3.2 Worries about Cloud Computing

The report written by Andrew Larkin points out most serious challenges that Cloud Computing are facing and offers some ideas to address these problems.[9] As mentioned before, safety and privacy are the keys of winning users' trust. However, hacker have successfully attacked the data centres in the past. Sensitive information and valuable technologies owned by corporations or private information and individual patents can not afford to be leaking. As a remote data centre, it is impossible for users to fix anything when problems occurred. It is a big disadvantage when compared to local data centre. Also, cloud servers can

be paralysed or even destroyed when natural catastrophe happens. This article uses various examples to present the difficulties when applying Cloud Computing and it keeps every point illustrated following a well-organized structure. Possible solutions are given to most of the problem together with explicit data to help readers getting better understanding. However, no reference is attached and many points such as accessibility and regulation.

Despite the bright side of this technology, there are also arguments and concerns against Cloud Computing. In the paper written by Sreeranga Rajan and Fujitsu, top ten challenges of big data security and privacy challenges are introduced and measures to enable safe and trustworthy cloud environments are explained. [13] The most important part is securing data storage and transaction logs. Unauthorized access and 24/7 availability are believed to be safer than Auto-tiering solutions. Validating and filtering input, applying real-time security and monitoring are being regarded as future developing domain. This is a simple guidance towards cloud security issues and it mainly just covered theoretical knowledge without practical examples.

4 Edge and Cloud Computing

The need for edge computing was grown from the success of IoT and rich cloud services[14, 15]. Edge computing and cloud computing are both essential architectures in our Internet today, especially at the fifth generation cellular networks era, in which network latency becomes an critical part [7].

Edge computing can be regarded as a post-cloud computing model[12]. According to the prediction from IDC, the number of apps at the edge will increase 800% [8]. We hereby discuss their relationship and difference in terms of goals and technical challenges.

4.1 Goals

Both edge computing and cloud computing share the same main purpose, to process and deliver data through network in different approach. Generally speaking, cloud computing is a centralized network where all data is handled in a central location, such as a data center. While edge computing is a decentralized distributed network that is close to the source of data.

Initially edge computing was born from content delivery networks (CDN), which provides web contents to end users as close as possible to make latency low [3]. The concept was later developed into edge servers, whose purposes were

similar to CDN. For example, Akamai deployed cloud computing near the end user to take the advantage of cloud computing and low latency [11]. Nowadays, edge computing performs computation on any functional computing entity at the edge, such as IoT devices. The goal is always same, close to users.

Edge computing and cloud computing are not mutually exclusive. While the decentralized edge computing provides benefits, the centralized cloud computing is needed to centralize the storage data and analysis of the high value business data [10]. After the data is processed at the edge, it will eventually reach the cloud, which acts as back-end access for central management. Edge computing needs cloud computing, especially in business. Additionally, computing power at edge devices is usually lower. If a large amount of data is involved in a computation, such as big data analytic and deep learning, the edge may not be able to handle them efficiently. Under such situation, the computation should be done in the cloud.

Cloud computing also needs edge computing. Cloud computing focus on providing on-demand availability of computer system resources. It provides many models for users to launch difference resources without the need of direct management of data center. It focuses on high-capacity, low-cost, and on-demand, but it did not consider the latency to end users at the first place. The cloud computing is deployed in data centers, and the latency depend on the physical distance and route paths between the end users and data centers. To solve the latency problem, companies deploy cloud computing in world-wide data centers and redirect end users to the closest data centers. In this case cloud computing is designated to edge servers [11]. Also, new architectural element such as cloudlet are designed to bring the cloud closer, which is also considered as edge computing [4].

4.2 Technical challenges

Challenges of traditional distributed systems such as scalability, reliability, efficiency, and security appear in a distributed cloud computing environment. The challenges also apply in edge computing because of its distributed nature. However, edge computing targets at tremendously large amount of unreliable things [15], there raise some new issues and challenges.

Scalability: Scalability is an important characteristic in any distributed systems. It requires the system to adjust the computing performance of each connected nodes to handle the increasing requests smoothly. It is very hard to measure accurately measure the scalibility of a system as it requires testing and

evaluating. The situation is worsen for edge computing because the amount of heterogeneity devices are dramatically increased. Some IoT devices like IP cams, mobile phones tend to have lower performance than cloud data centers. Edge computing should take these difference into consideration and have a more effective way of detecting the performance of the terminal devices.

Reliability: Reliability requires a distributed system to continue functioning when one or more nodes are unreachable. It is more complicated in edge computing. First of all, it may be hard to accurately identify the reason of failure of an IoT device, such as low battery, sensor error, or temperature controller goes down [15]. To mitigate the issues, it is necessary for the IoT devices to report detailed error information, and each IoT device must have the network topology of the whole system so that the system failure detection and recovery can be deployed easily. Moreover, the network connection of edge computing is mostly wireless, which tend to have unstable latency and packet loss at some time, it is not as good as cloud computing. This prevents the systems from delivering reliable service and pending solved with new wireless technology and algorithms.

Security: Encryption of data is a must in distributed cloud computing. In edge computing, because data is transit via multiple intermediate nodes before reaching the cloud, different encryption algorithms should be used according to security and resource requirements. Raw data should never be visible on the gateway of edge computing [15]. This involves in a new decentralized model that controls trust relationship in the edge [5].

5 Conclusion

It can be seen from this report that Edge Computing and Cloud Computing are not working alone separately. They have different focuses while being applied to solve complicated problems together. It is not wise to hold the ides of finding the so-called 'best' computing method or trying to tell the strict differences between them. Edge Computing can not replace Cloud Computing and vice versa. Edge computing aims to reduce the latency and thus place resources near the devices while Cloud Computing is designed to operate on big data. This explains the most important difference between them that Edge Computing is made for time-driven tasks while Cloud Computing is planned to analyse big data with a data center in a centralised location. However, there are more types of computing technologies are trending now, such as Big Data Analytics and Artificial Intelligence. More software are also being expected to thrive with the

composition of existing computing technologies to tackle specific problems and even bring up brand new ideas for modern industrial issues. The computing field is not isolated, together it works with software engineering, computer science and many other fields of researches to get improved. Cloud Computing and Edge Computing are both valuable technologies for the IT industry and business world. They are still considered has the potential of developing more applications to achieve higher demand tasks.

References

[1] Mehdi Bahram and Mukesh Singhal. "A Dynamic Cloud Computing Platform for eHealth Systems". In: (2015). [Online; accessed 20/05/2020].

[2] COMPAQ COMPUTER CORPORATION. "Internet Solutions Division Strategy for Cloud Computing". In: (November 14, 1996). [Online; accessed 20/05/2020].

[3] J. Dilley et al. "Globally distributed content delivery". In: *IEEE Internet Computing* 6.5 (2002), pp. 50–58.

[4] Elijah. *Cloudlet-based Edge Computing*. Accessed on May 24, 2020. URL: `http://elijah.cs.cmu.edu/`.

[5] Pedro Garcia Lopez et al. "Edge-Centric Computing: Vision and Challenges". In: *SIGCOMM Comput. Commun. Rev.* 45.5 (Sept. 2015), pp. 37–42. ISSN: 0146-4833.

[6] Eric Griffith. *What Is Cloud Computing?* `https://www.pcmag.com/news/what-is-cloud-computing`. [Online; accessed 20/05/2020]. 2016.

[7] Susan Welsh De Grimaldo. "AI + Edge Computing Essential in the 5G Era". In: *Strategy Analytics Service Providers Report* (2018).

[8] IDC. "IDC FutureScape: Worldwide IT Industry 2020 Predictions". In: (2019).

[9] ANDREW LARKIN. *Disadvantages of Cloud Computing*. `https://cloudacademy.com/blog/disadvantages-of-cloud-computing/`. [Online; accessed 20/05/2020]. AUGUST 7, 2019.

[10] David Linthicum. *Settling the edge computing vs. cloud computing debate*. 2019. URL: `https://www.infoworld.com/article/3435121/settling-the-edge-computing-vs-cloud-computing-debate.html`.

[11] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. "The Akamai Network: A Platform for High-Performance Internet Applications". In: *SIGOPS Oper. Syst. Rev.* 44.3 (Aug. 2010), pp. 2–19.

[12] Yi Pan, Parimala Thulasiraman, and Yingwei Wang. "Overview of Cloudlet, Fog Computing, Edge Computing, and Dew Computing". In: *Proceedings of The 3rd International Workshop on Dew Computing* (2018), pp. 20–23.

[13] Sreeranga Rajan and Fujitsu. "Expanded Top Ten Big Data Security and Privacy Challenges". In: (16 June 2013).

[14] W. Shi and S. Dustdar. "The Promise of Edge Computing". In: *Computer* 49.5 (2016), pp. 78–81.

[15] W. Shi et al. "Edge Computing: Vision and Challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646.

[16] Christof Weinhardt et al. "Cloud Computing – A Classification, Business Models, and Research Directions". In: *Business Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK* 1.5 (2009), pp. 391–399. URL: https://EconPapers.repec.org/RePEc:spr:binfse:v:1:y:2009:i:5:p:391-399.

[17] *What is cloud computing?* https://aws.amazon.com/what-is-cloud-computing/. [Online; accessed 20/05/2020].

Contribution

Futong Han	Introduction	Edge Computing
Binjie Zhou	Cloud Computing	Conclusion
Junyan Li	Abstract	Section 4 Edge and Cloud Computing
Group Work	Proofreading	Reference

Cloud Security Practices: Security Operations, Audits, and Compliances

Prashanth V. Dommaraju
(12998214)
University of Amsterdam
prashanthvarma.dommaraju@student.uva.nl

Neeraj Sathyan
(12938262)
University of Amsterdam
neeraj.sathyan@student.uva.nl

Vignesh Murugesan
(12970883)
University of Amsterdam
vignesh.murugesan@student.uva.nl

Abstract

Cloud computing has introduced a new age of information technology, with information comes privacy and security. History of data breaches and continuously evolving software and technologies add new types of vulnerabilities, security, and privacy issues. These issues affect cloud in a significant manner, mainly the public cloud service providers and cloud users. Few governments took proactive measures introducing policies such as the European union's General Data Protection Regulation GDPR. Security practices enable cloud users to defend themselves in case of breach or data leaks. Still, it is also the case of the cloud service provider's implementation of security, and there is always a tradeoff between choosing public providers and private providers. Compliances such as PCI-DSS, HIPAA, and ISO 27001 enable proper industry regulations. Public cloud service providers are also interested in government assets, and they are designing cloud services in that way. For example, AWS GOVCloud has different certifications and compliance requirements from the U.S. Federal government, allowing the government to safely move its assets to the cloud. We studied various surveys, studies, compliance, reports, models, and practices on cloud security. We provided our perspective on these studies with the most important security cases handled in the cloud sector nowadays.

1 Introduction

Cloud computing has been at the forefront of changing the I.T. infrastructure, as we know. Cloud computing made *Cloud Users*(CU) such as Enterprises and Organizations to efficiently reduce the cost, physical resources, and I.T. management; they still lack confidence in *Cloud Service Providers*(CSP). Cloud computing makes centralized management of data difficult. Evolving cloud computing and services introduced various types of security operations, audits, and compliance on *Cloud Service Providers*(CSP) to maintain integrity and trust with *Cloud Users*(CU). Our research question/assumption is how current *Cloud Users*(CU) and *Cloud Service Providers*(CSP) have their strategies, operations, the auditing process, and compliance's. International Data Corporation (IDC) conducted a survey (see Fig.1.) of 263 IT executives and their line-of-business colleagues to understand and see opinions on the I.T. cloud service usage. Security was found to be the prime issue of cloud computing.

4th Assignment on Web Services and Cloud Based Systems (UvA-5284WSCB6Y 2020), Amsterdam, The Netherlands.

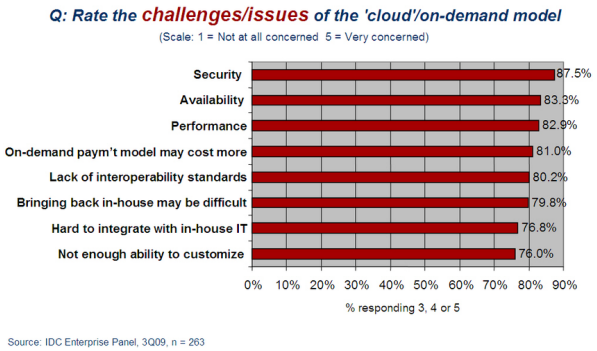


Figure 1: International Data Corporation (IDC) survey on On-Demand Cloud Challenges.(Source: [36])

Throughout the review, we considered the peer-reviewed papers and followed the compliance standards such as NIST and ISO/IEC. Reviewed the *Cloud Security Alliance*(CSA) research, such as Security Guidance v4 and Cloud Control Matrix. Security surveys also considered in the review. We selected papers, surveys, and articles based on the highest number of citations and year published from peer-reviewed journals such as *ScienceDirect*, *Institute of Electrical and Electronics Engineers (IEEE) Xplore* and *Springer*.

The 2nd section starts with the standard NIST definition of cloud computing and dives into different security models presented and implemented by the *Cloud Service Providers*(CSP). We considered the *NIST Cloud Multiple-Tenancy Model*, *The Cloud Risk Accumulation Model of CSA*, and *Jerico Forum's Cloud Cube Model* from paper [5] and discussed how different security issues affect cloud computing in this section. We also covered the *Third Part Auditor* part in security practices and discussed different papers proposed regarding privacy-preserving public audits [6],[7], and [8]. The 3rd section deals with the different compliances and regulations followed by the *Cloud Service Providers* such as *PCI-DSS*, *HIPAA*, *GDPR* and *IS0-27001*. This section also discusses about the *AWS GOVCloud* about how different government entities require different compliance and regulation requirement to earn there certification such as *FedRamp*, *NIST*, *SRG*, *FISMA*.

2 Security Models, Issues, and Practices in the Cloud

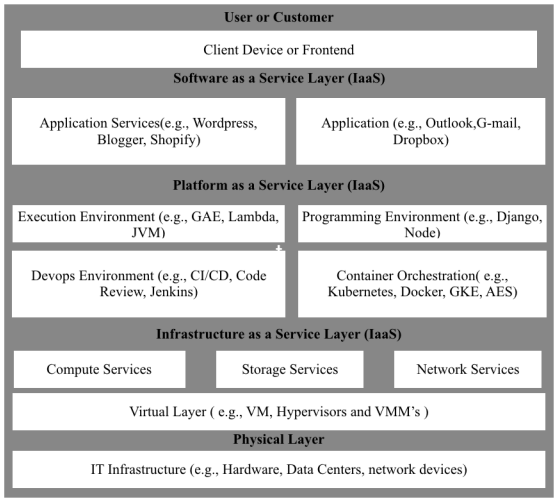


Figure 2: Cloud Service Deployment Models divided into Four categories, each containing their subcategories based on the NIST definition.

We considered the NIST definition of cloud computing. According to [1] *NIST Publication 800-145* the cloud has five essential characteristics, three service models, and four deployment models. We are going to reference these different characteristics and models throughout the review. We would have considered the ISO/IEC definition of cloud capabilities, but the NIST model is more publicly accessible and equally valid. [1] NIST model is widely adopted, and the definition is observed continuously in Cloud Security Alliance (CSA) research. The five essential characteristics are *On-demand self-service, Broad network access, Resource pooling, Rapid elasticity*, and *Measured service*. The three service models are *Software as a Service (SaaS), Platform as a Service (PaaS)*, and *Infrastructure as a Service (IaaS)*. Finally, the four deployments are *Private cloud, Community cloud, Public cloud*, and *Hybrid cloud*. Maximum no of available cloud services are a software as a Service built over Platform as a Service, which is built over Infrastructure as a Service.

The current trend of continually evolving cloud makes it difficult to point to an exact reference or architecture. However, according to the NIST definition and reference from the paper [2], we made this Fig. 2. for proper representation.

2.1 Security Models

2.1.1 NIST Cloud Multiple-Tenancy Model

[3] Clouds are multitenant by nature. Multiple different consumer constituencies share the same pool of resources but segregated and isolated from each other. Isolation allows *Cloud Users* to organize their assets better. Multi-tenancy nature does not apply to all the CU's, as some need more centralized management. [5] Virtualization allows different CU's to access the same physical resource such as processor, memory, storage, and I/O. As virtualization makes CU's isolated between VM's, this allows any malicious malware, virus, or attack to be isolated for that particular container. The difficulties of Multi-Tenancy models are [5] data isolation, architecture extension, configuration self-definition, and performance customization.

2.1.2 Jerico Formu's Cloud Cube Model

Jerico Formu's model is a graphical respresntation of security attributes in the cloud deployment models. There are four dimensions in this model as shown in the figure 3.

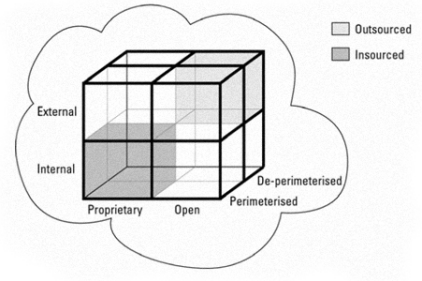


Figure 3: Jerico Formu's Cloud Cube Model.(Source: [9])

Internal/External: This represents the physical location of data, whether inside or outside of CU's limit. For example, the data on the public cloud is considered as the external and private cloud as internal.

Proprietary/Open: This parameter represents the interoperability of the data. Proprietary means that CSP's in control of all the cloud services such as private clouds. Where open means there is no lot of constraint on [5] data sharing and incorporation of businesses such as public clouds.

De-perimeterized/Perimeterized: *Perimeterized* means the data is running in a secure environment such as a private VPC or with a good firewall and IDS/IPS systems. *De-perimeterzied* system means that it does not have proper security mechanisms in place.

Insourced/Outsourced: [5] *Insourced* means that an organization's employees present cloud service, and *Outsourced* implies that a third party offers cloud service.

2.1.3 The Cloud Octagon Model of CSA

In this octagon model, Fig. 4., different sections represent different departments that are external/internal in an Organization.

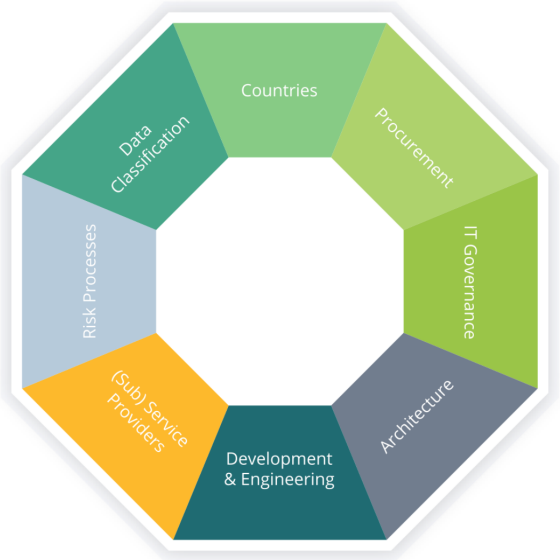


Figure 4: Cloud Octagon Model.(Source: [25])

[25] *Data Classification* section deals with whether the data on CSP has the following features: encryption, ownership, and management of data, logical access of data, and IAM Governance. *Service Providers, Development Engineering* and *Architechture* section deals with controls such as obtaining Assurance throughout the chain of providers, awareness of shared responsibilities between three different parties, and awareness of and mitigation in place for malicious insiders. *Procurment* ensures the right to audit, permission to conduct penetration testing, early risk identification, and mitigation. *Countries Using, Processing*, and *Hosting* section represents that data cannot be transferred to another country without consent, data in motion shall be protected, and compliance to laws and regulations in multiple countries. *IT, Governance, and Security Policies* section represents agreements on roles, responsibilities, and agreements on I.T. operating procedures.

2.2 Security Issues

Multi-tenancy also leads to the risk of the information or data visibility to other CU'S. Compromised physical layer or *Platform As A Serivce*(PaaS) layer can lead an attacker to take control of the *Software As A Serivce*(SaaS) or *Application As A Serivce*(SaaS) layer. One of the main characteristics of the Multi-tenancy is the Resource Pooling; this allows CU'S to share the computational, network, and storage resources. [10] The network resource sharing allows the cross-tenant attack as its hard to distinguish between a vulnerability scan and a real attack. Once if the attackers can get hands-on network components, they can launch attacks such as sniffing and spoofing. Security misconfigurations can comprise the security of the whole cloud system. For example, migration of V.M.s, data, and applications usually need a change of security policies. Any proper misconfiguration or weakness can lead to data compromise. CSP's also have data recovery mechanisms in case of damage or compromise; these features can also lead to security issues. Malicious users of CU's can access the previous user's data using data recovery. This study [13] revealed that over 5000 images on Amazon EC2 had problems such as the prevalence of malware, the quantity of sensitive data left on such images, and the privacy risks of sharing an image on the cloud. As most of the cloud controls are Web based UI's there are security issues which can impact the CU's such as [27] OWASP top ten web application security risks: *Injection*(e.g. Remote Code Execution), *Broken Authentication*(e.g. compromised passwords, keys, or session tokens), *Sensitive Data Exposure*(e.g. data such as financial, healthcare, and PII), *XML External Entities (XXE)*, *Broken Access Control*,

Security Misconfiguration, *Cross-Site Scripting*, *Insecure Deserialization* , *Using Components with Known Vulnerabilities*, and *Insufficient Logging and Monitoring.*

Security issues such as *Server Side Request Forgery*(SSRF) allow attackers to gain internal mechanisms of CSP's. For example, André Baptista reported an SSRF issue to Shopify's *Vulnerability Disclosure Program* on HackerOne. This report reveals that [11] it was possible to gain root access to any container in one particular subset by exploiting an SSRF bug in the screenshotting functionality of the Shopify Exchange application. According to the report, an attacker can gain metadata information such as SSH keys, CRT files, Network architecture, and Database keys, which can be the primary source of data leaks.

2.3 Security Practices

[26] CSA research and case study on the top threats regarding cloud computing revealed that human resource security could have acted as mitigation in six out of nine security breaches. While automated tools and scanners can perform suitable tasks in detecting vulnerabilities, there is a human factor that can be a reason for breaches. Proper security training can significantly decrease the attack surface layer for attackers. CSP's role is nevertheless low, and they should provide secure use of virtualization and proper images. This means CU's cannot run virtual images that are not authorized by CSP's. [3] CU's are also responsible for security regarding virtualization such as identity management, monitoring, logging, and image asset management. Container orchestration is widely adopted because it provides OS-level virtualization [28], where kernel allows the existence of multiple isolated user-space instances. [3] Containers do not offer total security isolation compared to virtualization but provide task segregation. Containers security is based on [3] deep understanding of O.S. internals, such as namespaces, network port mapping, memory, and storage access. CU's should understand the security isolation capabilities and operating system beneath it, [3] ensure only approved and secured images can be deployed, implement role-based ACL and strong authentication regarding managing the images and container repositories. CU's are typically tended towards *Third Party Auditors*(TPA) with experience and expertise, and they help to improve and assess their security based on TPA's report. However, TPA cannot be entirely trusted because audits can introduce new vulnerabilities and data leaks. These are a few papers [6],[7], and [8], which proposed the privacy-preserving audits. In paper [8], the authors discuss the public audit mechanism of implementing homomorphic authentication to verify the data blocks from CU's using metadata and adding a pseudo-random function to the linear combinations generated. Paper [6] proposes that CU's or *Data Owner*(DO) according to the paper, runs a KeyGen that generates *secret key*(sk) and *public key*(pk), where *sk* is pair of randomly generated key *spk* and key generated on signature of the filename *ssk*, where *pk* is combination of random generators and *spk*. Then the files are divide into blocks, and CU generates the authenticator for each block. They also proposed *doubly linked info table*(DLIT) and *location array*(LA) where the data, as explained in previous, is preprocessed to CSP so TPA can establish the DLIT and LA TPA runs a *Challenge Generation*(ChalGen) to CSP on behalf of CU and gets the file tag verifies with spk, then it picks up random DLIT blocks and sends the challenge. CSP generates tag proof and data proof to the challenge and sends it back to TPA. TPA then runs the VerifyProof to check the DLIT and LA data. TPA gets secure access to the CU's data. These privacy-preserving audits help CSP's and CU's to trust the TPA without any problem of data or information leak and protect the data integrity. Security practices consist of compliance and regulations too, but it is a topic of its own. Next section deals with the governance, compliance, and regulation in the cloud.

3 Compliance and Regulation in the cloud

An important aspect of cloud services when providing its customers with features and services is that they comply with the concerned laws and regulations to maximize protection from data leak and privacy issues. A non-compliance service provider can face severe penalties under the various laws governed under the cyber-laws of each country or union, where it operates or provides a service, and they become a target of a data breach or privacy leak. There are four major factors upon which the compliance certifications and rules for cloud security are based upon [7,14,15]:

- Data Protection
- Data Localisation
- Data Sovereignty

- Right to Information
- Inter-nations law and its context for application

Any new compliance standards evolve with these features as the fundamental model. Hence industry can try to standardize a cloud security model focused on all the above-said factors in order to reduce the complexity of introducing a compliance model certification. This can be a huge boost in saving humongous development and testing hours as, not only are there a lot of standards and regulations, but they constantly change, making it difficult for a cloud business to keep up with the ever-growing compliances and laws [13]. It should also be noted that the following compliances do not mean the CSP (Cloud Service Providers) are not susceptible to hacks. This is a misconception. The book [15] stresses that there is no guarantee of security, and compliance does not go hand in hand with security. Compliance is formed in order to regulate and make strict controls of cloud services that make it difficult for any security exploit, and even if a security breach happens, it can be dealt with further consequences. An example to discuss this common misconception is the common guidelines for setting strong passwords as in Fig.5. Setting up of strong passwords that follow certain lexical rules are now compliance in many organizations for their employees. However, this does not prevent a security breach like remote code execution or account takeover; the probability of the attack is lowered. *No system is safe* [16] is the standard guidelines the hacker community idolizes, and this tends to be true as new and weird security vulnerabilities keep on evolving based on the new upcoming cloud technologies. As more vulnerabilities surfaces, more and more compliance and regulations are put forth by the regulators and organizations. The literature [7,14,15,16] suggests an ever-growing tug-of-war between upcoming vulnerabilities and new compliance formation based on those vulnerabilities. This section provides a review of the history and the outlining goal of the compliance standards, cloud services follow, and its effect on an organization by selecting the most important compliance and standards currently being followed by cloud service providers. The problems identified within a majority of organizations in achieving compliance were found to be due to these reasons [14]:

- Auditing is not generalized for a virtualized distributed environment.
- Cloud infrastructure tends to be interleaved with different features and hence is difficult to match all of the compliance issues on all the cloud features.
- CSP Auditing may involve mandatory data share to the enforcing agency, including an NDA. The customers may not want even that to happen.

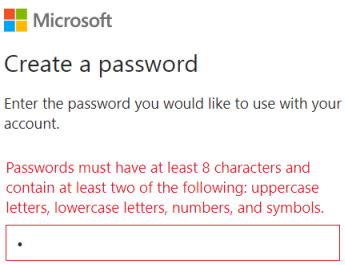


Figure 5: Mandatory Strong Passwords: Microsoft Outlook

The compliance regulators challenge Cloud Service Providers to establish, monitor, and demonstrate ongoing and applicable compliance rules with a set of controls that meet their customer's business and regulatory requirements. Maintaining separate compliance efforts for different regulations or standards is not sustainable, as previously thought of [3]. Hence organizations are focusing on a practical approach to audit and compliance in the cloud by distribution and coordination of internal policy compliance, regulatory compliance, and external auditing. Today, these three subsets make up almost all of the compliance types evolved by the regulators or the cloud organization. All of this should ultimately be beneficial to cloud customers, like:

- Protections against data breaches and leaks.
- Lower risk of legal sanctions.
- A good low cost per compliance implementation.

- Assurance of adherence to international privacy and security standards.
- Respect of rules of highly regulated industries like healthcare.
- Decreased business and financial risk in the long run.
- Easy handling of legal requests and audit processes.

3.1 Internal Policy Compliance

Internal compliances established within the organization providing cloud access or business can be completely private or open to the public. If a business makes the internal compliance, they reflect it upon the service level agreement to the CSP, to which the CSP has to comply [3]. Typically internal company level compliances are based on: *risk assessment*, *technology stack vulnerability and sustainable solutions*, *hierarchical developer access rights*, *information and communication*, *monitoring and solutions* [31]. Common and simple compliance that can be found in all tech companies (not only CSPs), is the merge-request permission before committing a production-ready code change to master branch in a distributed version control and source code management functionality tool like *git* [32], which will use *kubernetes* [33] to automatically deploy the master branch to production upon any new commit. These are not mandated or governed by external agents or customers but are crucially important in identifying and preventing potentially vulnerable code stack from running in production.

3.2 Regulatory/External Compliance

External Compliance requirements from both governing agencies and customers are more dynamic and keep on changing time after time. Keeping up with these compliances can make a cloud service boost customer satisfaction and business. Frequently, the requirements are based on or refer to industry standards. Hence CSPs can build a common framework using industry standards as a base model to work around the considerable quantity of standards and compliance models, as shown in Fig.6. However, only compliances related to information and cloud security will be discussed here to keep in scope with the article.

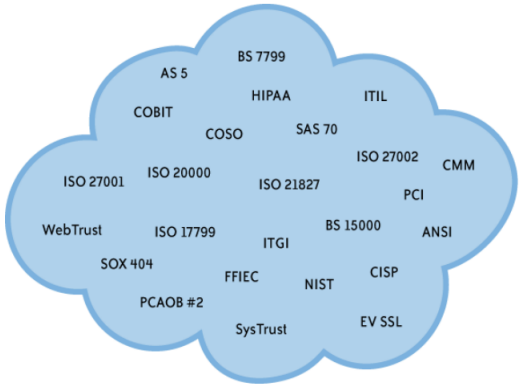


Figure 6: Ever growing regulatory compliances and standards

3.2.1 PCI DSS:

This is one of the first crucial compliance standards introduced in the cloud systems to maintain integrity and security of payment details. Now more than 53 countries mandate CSP to involve payment details to comply with this standard [37]. Online credit card and payment details are sensitive information that can lead to fraud or theft when exposed. To meet with these major payment gateway cloud companies like Visa and MasterCard have started their programs [29]. Validation and auditing are performed annually or quarterly by a method suited for the volume of transactions in records. Compliance is divided into seven groups of data and information security objectives:

- A secure and foolproof network and systems.

- protection and encryption of cardholder data.
- Maintaining a vulnerability management program.
- Implement SAC measures.
- Regular Monitoring and Testing.
- Maintain a proper information security policy.
- use of updated antivirus software certified within the industry.

The growth of multihop VPNs and advanced firewalls have made implementation easier over the years, although there are regular version updates of this compliance coming up [29]. This compliance, in itself, poses a sideline security threat. Logs and audit trails are often needed to investigate incidents, and organizations under PCI DSS have to provide timely forensic investigative data that can contain sensitive information to the regulators. The trust with regulators is the only solid line with which this compliance works. Nowadays, organizations are implementing end to end encryption of logs and introducing expiry dates. They are read-only access for these logs; hence once the auditors verify the logs, they are ready to be archived or deleted. An example of a breach under the PCI DSS would be to consider the Heartland Payment Systems (HPS) data security breach in January 2009. The company was audited and made PCI compliant two weeks before the breach. The breach was caused by a malware that was able to enter the payment services via SQL injection and cost the company millions in dollars of damages. This proves that being PCI compliant does not mean that systems are hack-proof. A rigorous pen testing methodology with all test cases must be involved in the system to prevent such attacks. These are primary everyday tasks needed to perform to test the integrity of the firewall and the organization's methodology that complies with PCI DSS.

3.2.2 HIPAA:

Health Insurance Portability and Accountability Act makes it mandatory for any cloud services to comply under its measures when supporting a business related to healthcare to get HIPAA certification. A risk analysis is the initial auditing process in this compliance step. Protected Health Information (PHI) is the target data focused on protection with this compliance, just like with credit card data for PCI DSS. This particular compliance was more in line with that of PCI DSS in terms of mandates and audits. However, the passing of the 2009 Health Information Technology for Economic and Clinical Health (HITECH) Act in the USA made significant changes to the compliance structure in terms of breach notification requirements, and accounting of disclosures of a patient's health information. With further developments in healthcare and with the rise of the medical health insurance companies that act as third party organizations to access the data stored within these HIPAA compliant CSPs, a lot of structural changes to the compliance model is expected in the future.

3.2.3 GDPR:

The GDPR (General Data Protection Regulation) is one of the pioneering compliance regulations put forth and mandated by the European Union against any organization that is involved with any business or involvement with its citizens. The core principle of GDPR is that **protection of personal data is a fundamental right**, and this regulation enforces it. The rights relating to the collection, storage, use, processing, transfer, etc. of personal data belonging to the individual, not data controllers, regardless of where data resides. If the company suffers a breach and the records of any E.U. citizen are compromised, the GDP jurisdiction will extend globally. That company may be pursued and fined significant sums of money. This new compliance is a huge undertaking for the implementation point of view that CSPs started separate dedicated teams to enforce this [18] gradually. The major difference between this compliance and others is, GDPR is a regulation and hence mandates that all positive organizations need to be GDPR compliant. GDPR became a model to design new privacy-related compliances outside E.U., including Chile, Japan, Brazil, South Korea, Argentina, and Kenya. The California Consumer Privacy Act (CCPA), adopted on 28 June 2018, has many similarities with GDPR [19]. GDPR focuses on a secure flow of personal data, and its rights belong to the person to which the data belong. GDPR is implemented and auditing is carried out using two divisions: *data controller* and *data processors*. Some of the rights of a data subject who has his/her data stored within those CSPs are:

- Right to require consent (and where applicable, to withdraw consent).

- Right to request that businesses delete the individual's personal data(right to be forgotten).
- Right to access their personal data (subject access).
- Right to have inaccuracies corrected (rectification).
- Right to data portability.
- Right to object to certain uses (e.g., processing, direct marketing, profiling).
- Right to complain to supervisory authorities.

3.3 Internal/External Auditing and Standards

3.3.1 ISO 27001:

ISO/IEC 27001 defines mandatory requirements for an Information Security Management System (ISMS). Its a certification given for CSP dedicated to the field of information system security and web services interoperability [20]. This standard provides a blueprint to provide a model for: *establishing*, *implementing*, *operating*, *monitoring*, *reviewing*, *maintaining*, and *improving* information security management system.

3.4 Gov Cloud Certifications for Public Clouds

Public Clouds run by governments are increasingly becoming popular to register and process different civil conducts like tax filing, municipal rent agreement, etc. As more and more government intermediary services are becoming online, public clouds start providing services to it. Thus comes a challenge to protect the citizens and the government's sensitive data from a breach as well as from other government agencies that can mainly utilize their domestic law to eavesdrop on the data used by different countries in the CSP hosted in that country's location. An example of this can be the case of the USA's Patriot Act, which allows the U.S. government power to retrieve and analyze data from the servers hosted by CSPs at their sovereign location. This prevents significant government agencies outside the USA to trust any USA registered CSPs with their internal data and have a higher demand to implement its public cloud [21]. As the concerns of government data leaks are a primary concern within public clouds, different gov cloud certifications have evolved to tackle these issues in a more step-up manner. Some of the most common certifications used today in public clouds are FISMA, NIST, SRG, and FedRAMP.

3.4.1 FISMA

The Federal Information Security Management Act of 2002 (FISMA) considers information security as the topmost priority in a public cloud. It requires U.S. agency heads to implement policies and procedures in a cost-effective process to alleviate security risk to a minimum. It mandates that sensitive data is not shared outside of the country to become FISMA certified and provide cloud services to the government. The core concept of FISMA is to make stringent rules as to make a cloud service provider limited to data sharing capability and make it easy for the U.S. government to utilize those cloud services. The Act was updated recently in 2014 to the Federal Information Security Modernization Act.

3.4.2 NIST

The National Institute of Standards and Technology (NIST) is a non-regulatory agency belonging to the Department of Commerce of the United States Government. This agency attributes standards and government compliances to public cloud service providers to enhance the data security within it. NIST creates and promotes information security standards for the federal government, which then indirectly applies to the CSPs in agreement with the government and must comply with them.

3.4.3 SRG

United State's Department of Defence (DoD) Cloud Computing *Security Requirements Guide* provides authenticity to a CSP so that defense and military data, specifically intelligence data, can be stored and analyzed using those certified infrastructures. The DoD Cloud Computing SRG leverages the FedRAMP program to establish a standardized approach for the DoD to assess cloud service providers (CSPs).

3.4.4 FedRAMP

The Federal Risk and Authorization Management Program (FedRAMP) made possible the U.S. federal government's *cloud-first* initiative by making the transition of signing contracts with CSPs to make business easier. FedRAMP, unlike FISMA, is specific only to cloud service providers. Therefore its auditing and certification process is far more stringent and rigorous in terms of cloud computing security standards. Most private-sector cloud companies recognize this as the gold standard for certification in cloud security.

a good point to be noted all the servers of AWS under the compliance of FISMA, NIST, SRG, and FedRAMP all are stored and made multiple instances right inside the U.S. territory.

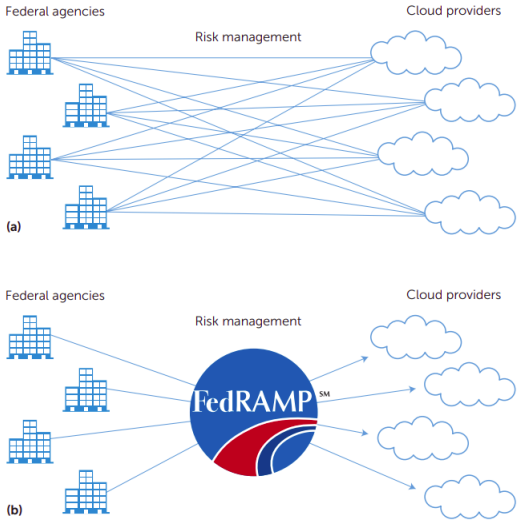


Figure 7: Authorization process for federal clouds: (a) old way and (b) new way. (Source: [23])

4 Conclusion

Cloud computing can offer many advantages to the CU's, but security is the primary reason CU's find it challenging to adopt. Where startups, small, and medium-size companies are more flexible towards cloud because of the economic feasibility. Government and entities with highly confidential data are hesitant to shift all their assets to the cloud. Strict compliance requirements are progressed to meet those requirements. However, there is no guarantee that compliance and regulations can stop breaches. For example, AWS SOC 3 report [35] states that *Vulnerabilities in information technology components as a result of design by their manufacturer or developer* and *sophisticated social engineering techniques specifically targeting the entity*. This indicates that CSP's cannot provide total security to CU'S because CSP's job is to ensure and provide proper security to their IaaS and PaaS platforms. As critical controls are given to CU's, some of the responsibility also belongs to them. [34] Synack(Crowd-sourced pen-testing company established by former NSA employees and has significant fortune 500 companies as clients) conducted a broad survey on more than 311 organizations from the USA, where there were 25% technological and 17% government organizations. The study found some compelling aspects: 54% of organizations follow internal compliance and security standards. 17.3% of organizations have implemented GDPR since it is inception before two years. 63% use external pen-testing for identifying the vulnerabilities. The reason for more external testing was to meet the compliance requirements and reduce vulnerabilities. As cloud services are expanding and integrating new features, it also increases the attack surface layer and new compliance and regulations. *Artificial Intelligence* can be prominent in defensive cloud security practices. Future security practices also require regulations for A.I./ Machine Learning(ML) systems. Proactive measures can lead to better safety and privacy for the upcoming generation of humans.

References

[1] P. Mell, T. Grance The NIST Definition of Cloud Computing. *Special Publication 800-145 NIST* (2011) https://doi.org/10.6028/NIST.SP.800-145 .

[2] Fernandes, Diogo & Soares, Liliana & Gomes, João & Freire, Mario & Inácio, Pedro. (2013). Security Issues in Cloud Environments - A Survey. *International Journal of Information Security: Security in Cloud Computing. 10.1007/s10207-013-0208-7.*

[3] Cloud Security Alliance. *Security guidance for critical areas of focus in cloud computing(v4.0).* Decemeber, 2009. https://downloads.cloudsecurityalliance.org/assets/research/security-guidance/security-guidance-v4-FINAL.pdf

[4] Jansen, Wayne A., and Tim Grance. Guidelines on security and privacy in public cloud computing. (2011).

[5] Che, Jianhua & Duan, Yamin & Zhang, Tao & Fan, Jie. (2011). study on the Security Models and Strategies of Cloud Computing. *Procedia Engineering. 23. 586–593. 10.1016/j.proeng.2011.11.2551.*

[6] J. Shen, J. Shen, X. Chen, X. Huang and W. Susilo, An Efficient Public Auditing Protocol With Novel Dynamic Structure for Cloud Data," in *IEEE Transactions on Information Forensics and Security, vol. 12, no. 10, pp. 2402-2415, Oct. 2017, doi: 10.1109/TIFS.2017.2705620.*

[7] Lins, Sebastian & Schneider, Stephan & Sunyaev, Ali. (2018). Trust is Good, Control is Better: Creating Secure Clouds by Continuous Auditing. *IEEE Transactions on Cloud Computing. 6. 1-14. 10.1109/TCC.2016.2522411.*

[8] C. Wang, Q. Wang, K. Ren and W. Lou, Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing, 2010 *Proceedings IEEE INFOCOM, San Diego, CA, 2010, pp. 1-9, doi: 10.1109/INFCOM.2010.5462173.*

[9] Jericho Forum's Cloud Cube Model, https://www.w3schools.in/cloud-computing/cloud-cube-model/, Last accessed 27-05-2020.

[10] Ali, Mazhar & Khan, Samee & Vasilakos, Athanasios. (2015). Security in Cloud Computing: Opportunities and Challenges. *Information Sciences. 305. 10.1016/j.ins.2015.01.025.*

[11] André Baptista, SSRF in Exchange leads to ROOT access in all instances. https://hackerone.com/reports/341876, Last accessed on 27-05-2020.

[12] Marco Balduzzi, Jonas Zaddach, Davide Balzarotti, Engin Kirda, and Sergio Loureiro. 2012. A security analysis of amazon's elastic compute cloud service. *In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12). Association for Computing Machinery, New York, NY, USA, 1427–1434.* DOI: https://doi.org/10.1145/2245276.2232005.

[13] An Introduction to Cloud Computing for Legal and Compliance Professionals, Microsoft Cloud. https://download.microsoft.com/download/0/D/6/0D68AE95-6414-4074-B4B8-34039831E2BF/Introduction-to-Cloud-Computing-for-Legal-and-Compliance-Professionals.pdf, Last accessed: 27-05-2020.

[14] Baldwin, Adrian, Simon Shiu, and Yolanta Beres. Auditing in shared virtualized environments. *Hewlett-Packard Labs Technical Reports 4* (2008): 2-19.

[15] B. R. Williams, A. Chuvakin. PCI Compliance: Understand and Implement Effective PCI Data Security Standard Compliance, 4th Edition, 2015 ELSEVIER ISBN: 978-0-12-801579-7

[16] Cole, Eric. Advanced persistent threat: understanding the danger and how to protect your organization. Newnes, 2012.

[17] Elluri, Lavanya, and Karuna Pande Joshi. A knowledge representation of cloud data controls for eu gdpr compliance, 2018 *IEEE World Congress on Services (SERVICES). IEEE, 2018.*

[18] Duncan, Bob. Can eu general data protection regulation compliance be achieved when using cloud computing?, *Cloud Computing 2018: The Ninth International Conference on Cloud Computing, GRIDs, and Virtualization.* IARIA, 2018.

[19] General Data Protection Regulation https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, Last accessed on 22-05-2020.

[20] Vines, Ronald L. Krutz Russell Dean, and R. L. Krutz. Cloud security: A comprehensive guide to secure cloud computing. Wiley Publishing, Inc, 2010.

[21] Diez, Oscar, and Andres SIlva. Govcloud: Using cloud computing in public organizations. *IEEE technology and society magazine 32.1 (2013): 66-72.*

[22] Mell, Peter, and Tim Grance. The NIST definition of cloud computing. (2011).

[23] Taylor, Laura. FedRAMP: History and future direction. *IEEE Cloud Computing 1.3 (2014): 10-14*.

[24] Omotunde, Ayokunle & Oludele, Awodele & Kuyoro, Shade & Chigozirim, Ajaegbu. (2013). Survey of Cloud Computing Issues at Implementation Level. *Journal of Emerging Trends in Computing and Information Sciences. 4. 91=96*.

[25] CSA Cloud Octagon Model, `https://cloudsecurityalliance.org/artifacts/cloud-octagon-model`, Last accessed on 22-05-2020.

[26] Top Threats to Cloud Computing: Deep Dive. Release Date: 08/08/2018. `https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-deep-dive/`, Last accessed on 22-05-2020.

[27] Owasp Top Ten. Top 10 Web Application Security Risks. `https://owasp.org/www-project-top-ten/`, Last accessed on 22-05-2020.

[28] OS-level virtualization, `https://en.wikipedia.org/wiki/OS-level-virtualization/`, Last accessed on 22-05-2020.

[29] Morse, Edward A., and Vasant Raval. PCI DSS: Payment card industry data security standards in context. *Computer Law & Security Review 24.6 (2008): 540-554*.

[30] Bailey, Sarah F., et al. Secure and robust cloud computing for high-throughput forensic microsatellite sequence analysis and databasing, *Forensic Science International: Genetics 31 (2017): 40-47*.

[31] Graham, Lynford. Internal control audit and compliance: documentation and testing under the new COSO framework. *John Wiley & Sons*, 2015.

[32] Git, `https://en.wikipedia.org/wiki/Git` . Last accessed on 22-05-2020

[33] What is Kubernetes?, `https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/` . Last accessed on 22-05-2020

[34] The 2020 state of compliance and security testing report, Synack, `https://www.synack.com/resources/the-2020-state-of-compliance-and-security-testing-report/`, Last accessed on 27-06-2020.

[35] AWS SOC 3 – Security & Availability, New SOC 1, 2, and 3 Reports Available, `https://aws.amazon.com/blogs/security/new-soc-1-2-and-3-reports-available-including-a-new-region-and-service-in-scope/` Last accessed on 01-06-2020.

[36] Enhanced Cloud Security by Combining Virtualization and Policy Monitoring Techniques - Scientific Figure on ResearchGate. Available from: `https://www.researchgate.net/figure/Cloud-Challenges-Issues-by-IDC-Survey-The-fig1-shows-the-cloud-challenges-issues-a_fig1_271617323` Last accessed on 01-06-2020.

[37] PCI Compliance is Mandatory, `https://www.newnettechnologies.com/pci-compliance-is-mandatory.html` , Last accessed on 31-05-2020.

Contributions:

Team Contribution Split Table			
Group Member	Tasks	Details	Duration
Prashanth V.Dommaraju (12998214)	Introduction, Security Models	Involved in collecting appropriate peer reviewed literature and industry standard websites, gathering points, analysing papers, retrieving insights from each papers, latex report writing	8-9 days
Neeraj Sathyan (12938262)	Compliances, external compliances	Involved in collecting appropriate peer reviewd literature and industry standard websites, gathering points, analysing papers to see the trends in the usage of compliances over CSPs, latex report writing	8 days
Vignesh Murugesan (12970883)	Internal compliance and audit processing	Fact checking, analysing papers to get audit controls and internal company compliance standards, proof reading, error correction, latex report writing	8 days

High Performance Computing in The Cloud (Group 17) : Literature Study

Octafarras, Gamma A.
me@gammaandhika.com

Radhakrishnan, Abijith
mail@abijith.net

Singh, Tavneet
t.s.tavneet@student.vu.nl

Abstract

High-Performance Computing (HPC) workloads have traditionally been run on private clouds and on-premise clusters. Until recently, the latency, security and lack of customizability of public clouds have made them an unattractive option for running HPC workloads. But with newer services and abstractions, the cloud service providers have claimed that public cloud is a viable option with lower costs, ease of scale, higher customizability while guaranteeing QoS and SLAs. This paper looks at the validity of the claim and challenges of public clouds being used for HPC workloads of scientific and business applications through a survey of research papers and whitepapers. Different facets of HPC applications in terms of performance, cost and security in both public and private clouds are analysed and compared before presenting a comprehensive discussion and conclusion of our work. The potential for HPC applications in the cloud is promising especially due to the major innovations made by cloud service providers in the past years to bridge the gap separating private and public clouds.

Contents

1 Introduction

High Performance Computing most commonly associates to the method of assembling computing capability in a way that produces much higher performance than a regular desktop computer or workstation in order to tackle large problems in science, engineering, or business and gives the ability to process data and perform complex calculations at high speeds. The acronym HPC can mean both high-performance computing or high-performance computers, depending on the context.

A supercomputer is the most common type of HPC solution and it usually consists of 100s of compute nodes working together to finish one or more tasks. This is termed as parallel-processing and it is comparable to having a large number of PCs networked together to merge their computing power for quicker task completion.

HPC brings together multiple technologies including programming, architecture, system software and algorithms, working together to produce swift and efficient results. HPC systems then use computing resources simultaneously to achieve robust performance.

HPC applications are being employed for different purposes across multiple industries; from being used in research labs to help scientists tackle issues such as weather prediction, renewable energy and astrophysics to financial services to track stock exchange trends. Furthermore, HPC is widely used in the field of artificial intelligence and machine learning. It is also effective in product design as well as to test simulations.

The first incitation for high performance computing transpired with the conception of cluster computing. Cluster computers are loosely linked parallel computing device where the computing nodes have separate memory and operating system instance, but usually with a shared file-system, and employ an explicitly programmed high-speed network for communication.

Cluster computing connects many computers in a network and then work like a single entity. A computer that is a part of this cluster computing network is called a node. Standard cluster computing is devised to create a repetitive environment that will guarantee an application will continue execution despite a hardware or software malfunction.

Cluster computing further evolved to form grid computing. Grid computing follows a distributed method for solving computation heavy problems that can not be done with the common cluster computing design. Instead of replicating computation device and its corresponding setting to create a redundant environment, a grid computing cluster is a group of computers loosely coupled to function as independent modules or solve independent problems. Grid computing is devised to run independent tasks in parallel, thereby leveraging the computer processing power of a distributed model.

Cloud computing emerged with a similar goal as grid computing - to provide a service where computing resources can be shared with multiple users, but focusing more on the aspects of flexibility, availability and accessibility. Cloud computing comprises of hardware as well as software resources made accessible over the internet as external services. Also, cloud service providers build cloud computing servers to provide prevalent business and research requirements. A private cloud is a cloud computation service that is not used by any other organization and the user or organization has the cloud to themselves. By comparison, a public cloud is a cloud computing service that allocates computing services between several users and the services are provided over the internet.

1.1 Motivation

HPC in the Cloud is an attractive option because the end-user can obtain on-demand computing capacities. This additionally provides elasticity as well as workload resilience to handle sudden spikes in computation. Furthermore, compute resources are calculated at a granular level, allowing customers to spend only for the computations they use, potentially reducing operational costs. The

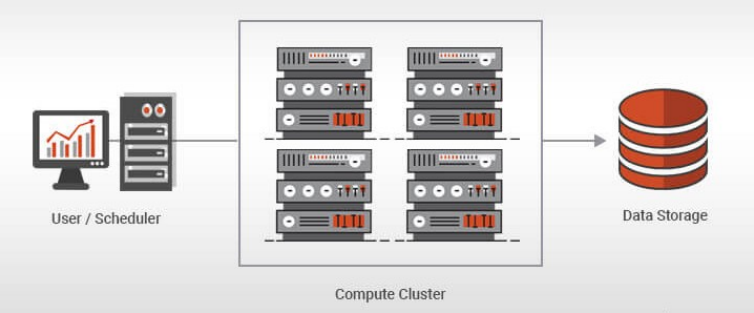


Figure 1: High Performance Computing Architecture

cloud providers can provide access to the latest hardware through vendor deals and frequent addition of new hardware to their data centers (more frequently than a private cloud).

In the subsequent sections, different cloud computing services are analysed to answer the question - *Is the public cloud a better alternative for high performance computing than private clouds*? After presenting an overview, and an in-depth analysis of major challenges of HPC on cloud computing, the current state and adoption of HPC on the cloud are discussed before concluding the paper.

2 Overview of Cloud Computing

This section delves deeper into the technicalities of cloud computing and also explore the current state of HPC in the cloud.

2.1 Cloud Computing

Initially, cloud computing emerged as a solution for business applications. A platform where businesses could build their web applications on without having to own a traditional server, with one of the first major public cloud platform released in 2006 by Amazon as EC2. In recent years, cloud computing has become even more relevant since the advancement of hardware virtualization technologies and the concept of micro-services. In which the use of cloud computing has become a necessity for new businesses.

The cloud computing technology itself is built on a concept that is similar to grid computing, where multiple compute resources are used together to perform tasks. Though in the scope of cloud computing, resources in the form of data centers are used instead of distributed clusters, where multiple computing resource might be located in different geographical locations. Cloud providers generally use an abstraction where the provider itself would control the hardware and virtualization whereas the user would have access up to the OS layer 2.

As cloud technologies evolve, major cloud providers have been branching out to not only provide Infastructure-as-a-Service(IaaS) for businesses, but also multiple fields of computing. Of which in the scope of this review will be the use of HPC in the cloud.

Cloud computing is now commonly split into three types - public, private and hybrid clouds. Additionally, community clouds also exits but is out of the scope of this paper.

2.1.1 Public Cloud

A public cloud is the most commonly used platform for cloud computing. Where the computing resources are hosted by third party provider and is accessed through the internet. The resources are also shared by multiple users, in which individual VMs are built on top of shared hardware.

2.1.2 Private Cloud

Private cloud, in a technological view, is identical to the public cloud. The main difference being that private clouds dedicate hardware and software to a single user, which in most cases, but not limited to,

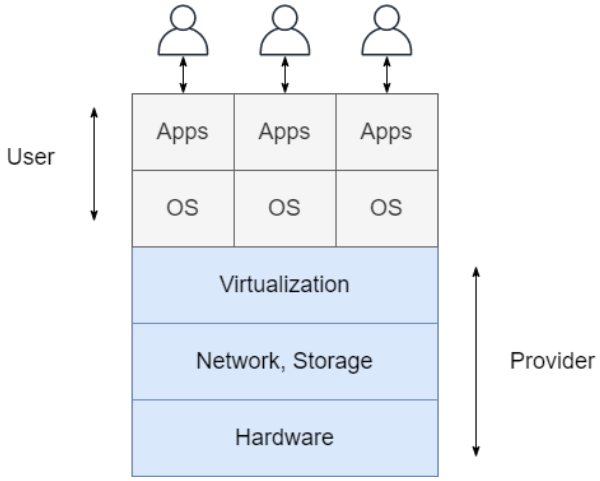


Figure 2: General Public Cloud Abstraction

are medium-to-large organizations. The data center of a private cloud may be located in the premise of the organization or be hosted by a data center elsewhere. Resources are not shared between other users, hence the improvement of security and customizability which is a factor to why the cost of private clouds are generally higher than public clouds. Additionally, in this paper we will refer to supercomputers and HPC cluster as part of the private cloud.

2.1.3 Hybrid Cloud

Hybrid clouds have been gaining more interest over the years, and is generally the solution for users looking for a balance between the features of public and private clouds. Hybrid clouds provide the flexibility and cost efficiency of a public cloud while still retaining the control and security of a private cloud.

2.2 Current state of HPC in the cloud

Early research suggested that though cloud computing can be a viable platform to run HPC applications, there are still several challenges that needs to be resolved. The main challenges for the adoption of HPC in the cloud are - security, performance and cost.

An early case study conducted by He et al. (2010) [27] which aims to test performance of running a NASA climate prediction app in the cloud, shows that performance was hindered the by the slower network speeds of the cloud instance that was used. These challenges has since been contested by major public cloud providers, of which examples would be Amazon Web Services(AWS), Google Cloud Platform(GCP) and Microsoft Azure. Amazon claims that their AWS platform has solved the challenges of running HPC applications in the cloud [16], and provides instances that are tailored to run HPC apps [9].

The 3 challenges that have been mentioned will be further explored in the next section.

3 Comparison

3.1 Performance

Performance evaluation of high performance computing in the cloud has been a popular topic as earlier studies [20, 41, 33] concluded that using public cloud for tightly coupled HPC applications led to a performance degradation. The lower performance was attributed to networking latencies caused by commodity interconnects like 1 GBPs Gigabit, processor sharing and virtualisation I/O overhead [23].

The challenges led to academic research to benchmark and suggest new placement strategies, optimisations to improve performance on their private clusters and public clouds.

A way to improve performance was looked into by Gupta et al. [24] where they suggested making the cloud HPC aware to reduce communication overhead by using containers, thin VMs and taking advantage of CPU affinity by binding virtual CPUs to physical CPUs. They also presented a simultaneous approach of making HPC applications cloud aware by tuning the problem and granularity size to reduce communication across nodes. They concluded that public clouds showed promise for small to medium scale startups, research groups which did not have access to a supercomputer and could be used in a hybrid setting to handle burstiness for groups that did. Their results also showed that virtualisation overhead was low (1-5%) and Embarrassingly parallel applications not involving high communication amongst the nodes scale quite effectively.

In another study [25], Gupta et al. looked into making VM placement HPC aware for performance gains by using homogeneous VMs and an application characteristics(cache, communication behaviour)-aware scheduler. Jin et al. [29] deployed a cache contention-aware(CCAP) VM placement approach for HPC. Both the scheduler and CCAP improve the performance of the in-cluster VMs. So the research pointed to having separate HPC VMs or handling HPC application differently from the standard cloud VMs being used for IaaS and PaaS services.

To counter the networking limitations, public cloud providers introduced new VM types targeted for HPC having full bisection bandwidth (AWS Cluster Compute [1]), using infiniband interconnect (Azure HB Series [8]), Azure H Series [2] and compute optimised VMs C2 on GCP[6].

Studies were conducted to compare these HPC optimised VMs amongst each other and with the in house clusters. Robert et al.[23] compared the Amazon Cluster Compute - CC1 and CC2 instances. Their results showed that para-virtualisation of the NICs hindered scalability and CC2 instances despite having higher computational power performed worse than CC1 instances for collective-based communication intensive applications like Integer Sort and Fourier Transform.

Hassani et al.[26] compared a parallel optimised version of Radix Sort on their in-house with the AWS EC2 cluster with AWS cluster showing better execution times. Although, the cluster size was small in this study (8 nodes).

Single Root I/O Virtualisation (SRV-IOV) is a PCIe technology which reduces I/O overhead in virtualised environments through virtual functions and direct access to the hardware [39] [34]. Combining SRV-IO with RDMA networking like InfiBand interconnect has been shown to improve HPC cloud performance [32] [42]. Kotas et al. [31] did a comparison between Azure and AWS HPC-optimised instances having SR-IOV technology and 10 GB interconnect over for HPCG/ PTRANS benchmark and concluded that Azure instances offered a faster solution for communication intensive applications.

A study evaluating performance of a private Fermicloud vs Amazon EC2 instances [37] showed that communication overhead was still a bottleneck for scientific applications in the cloud leading to lower performance. AWS instances had higher compute performance, while Azure instances had higher network performance. The performance bottleneck was also mentioned by [10].

In contrast, a comparison of public cloud providers by Mohammadi et al.[18] using High Performance Linpack [3] where Microsoft Azure showed the highest speedup because of InfiniBand interconnect across its H series VMs. A comparison of NERSC Edison supercomputer [5] with the public cloud provider HPC optimised VMs showed a better performance per GigaFlops by some of the VMs like Azure H series, Azure A Series, Amazon c4.8xlarge instances, etc as show in Figure 3.

Public cloud providers have recently pushed into efforts for HPC. There were various blog entries in 2019 announcing new products for HPC like Cray Cluster Store in Azure (a parallel storage platform based on Crays ClusterStor[4])[2], Azure HPC Cache [28] (a cache containing active data located both on premise and cloud to reduce latencies), GCP DDN's EXAScaler (a parallel file system based on Lustre [5] to handle high concurrency access patterns to shared data sets)[15]. The studies

[1] https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html

[2] https://docs.microsoft.com/en-us/azure/virtual-machines/h-series

[3] https://top500.org/project/linpack/

[4] https://www.cray.com/products/storage/clusterstor

[5] http://lustre.org/

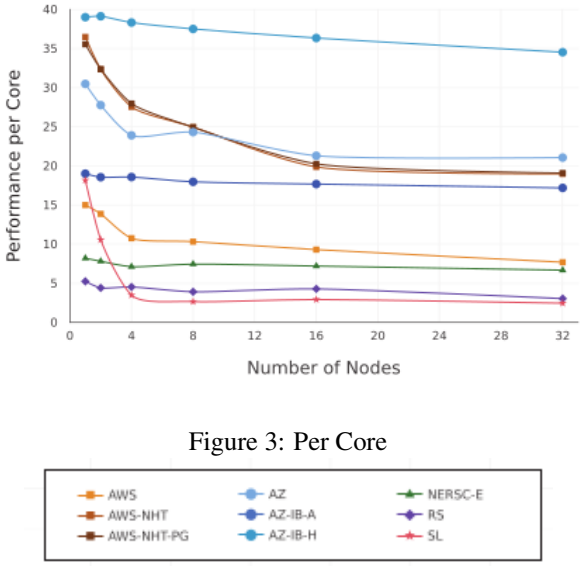


Figure 3: Per Core

show that the performance limitations presented by the public virtualized cloud environment have reduced considerably over time. HPC seems to be a major focus of cloud providers with Hyperion research[14] showing a major trend towards increased adoption of public cloud amongst enterprises and predicting a growing trend.

3.2 Cost

Potential cost savings benefits exist as the cloud offers a pay as you go model based on the utilisation rate. The total cost of operations for an on-premise setup include the capex(setup) and opex(operational) expenses while the cloud only has opex. The cost benefits are one of the major driving factors for cloud usage for IaaS instances. [1] [30]. Doing a cloud comparison is challenging because the in-house cluster nodes generally do not map directly with the Cloud instance providers. The dynamic pricing and ever evolving VMs have led to less academic literature comparing the cost efficacy.

Roloff et al. [36] used a cost efficiency per hour model to compare RackSpace, Azure, AWS and their cluster instances and defined a break even point (the yearly utilisation days required) where it became more cost efficient to use an in cluster setup. From their results, the cluster was more cost efficient when utilising a single node but Azure performed better than the cluster when the application was run on 4 nodes. The cloud was also a viable option in case of a low utilisation rate (less than 50% for Azure and less than 25% by AWS).

Emeras et al. [21] came up with a Total Cost of Ownership Model (TCO) to compare their on-premise HPC setup with AWS instances. Their results showed that some instances had comparable hourly performance with closest related configuration AWS instances while the high memory VMs did not. After adding up the costs for storage, they presented that the in-house solution was more cost effective than the public cloud instances. Prabhakaran et al. [35] used the same TCO model to compare their in-house supercomputer with AWS, Azure and GCP with the outcome of public cloud estimated to be 3-6 times more expensive. The lower cost for in-house setup was attributed to high utilisation rates, low manpower costs for maintenance and academic pricing of the hardware for in-house setup.

Another aspect to evaluate the cost in the cloud is the egress cost. It is costly to migrate data from a cloud vendor leading to vendor lock-in. [3, 12]

The number of nodes in [36] was less (1,2 and 4) and the authors didn't provide a comprehensive breakdown of the costs as compared to [21] and [35]. From these studies, it is more cost effective to run nodes with high utilisation on an in-house cluster. But differential pricing and further market investments could help drive the price down with time. Evaluation of spot markets to reduce the cost could be a potential research area.

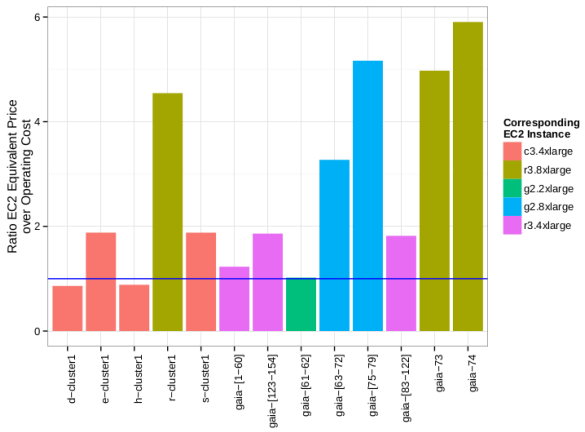


Figure 4: Luxemborg Instances

3.3 Security

Security became a challenge that was slowing down the adoption of HPC in the cloud. An early study by Hill et al. (2008) [22] states that running HPC applications in a public cloud introduces security issues which mainly stems from the fact that nodes are hosted by a third-party and that access is through the internet. They also mention that there is a possibility of an attack incited by another user on the same node. Other main security concerns revolved around the issue of access control management and data privacy. From the problems that have been stated, it was observed that HPC in the cloud itself does not add more security complications and the concerns are is inherited from the security problems of cloud computing itself.

With the advancement of privacy laws and regulations such as the General Data Protection Regulation (GDPR), previous concerns on data privacy has lessened. Major public providers also claim that since the advancement of security tools such as Identity and Access Management for AWS, access control is now secure. For the concerns that were brought up by Hill et al., a hybrid cloud is a viable solution. A hybrid cloud would considerably lessen security burdens as you can run sensitive data in the private cloud side, but still leverage the public cloud for supplementary computing needs.

Though interest in the field of cloud security has been steadily rising since 2008 [7], research specific to HPC cloud security is sparse and security is rarely mentioned in recent HPC cloud papers. Even though open cloud security issues still exist, the general notion seems to be that the benefits outweigh them.

4 Discussion

Most of the fascination for cloud computing is due to convenience, cost and flexibility. However, the issues faced by HPC users in the cloud are more complicated compared to an average cloud user. In this section, we discuss a brief summary of arguments and counter arguments for achieving HPC using public cloud.

After an extensive study of the given topic, it can be concluded that cloud computing systems have evolved enough to make them appealing for HPC users. Irrespective of being public, private or hybrid cloud, they offer unparalleled versatility for users to add nodes with distinct architectural specifications, apply cloud bursting to expand the capacity of the infrastructure.

Even though the system administrators are alleviated from the obligation to maintain expensive physical hardware, they still have to manage different networking, software and frameworks used in the complex, modern HPC problems. Also, more responsibilities such as managing cloud credentials, permissions, security, data synchronization are added. As on-premise server management tools evolve, there's a case to be addressed that operating an HPC in the public cloud could be every bit as intricate as maintaining a local compute cluster.

Despite the fact that cloud services have evolved a lot, it is not necessary for cloud-instances to be quicker than on-premise servers. It is worth noting that a cloud vCPU is not a physical core but threads in a hyper-threaded core. A non-peer reviewed study observed that the users need to provision around 27% more vCPUs in the cloud to achieve similar computing capacity in an on-premise setup [4].

Considering the cloud has a pay-as-you-go subscription model, the cloud can be regarded as the cheaper option for heavy work-loads of short duration. But heavy work-loads of short duration is uncommon for an HPC data centre compared to a regular cloud user with low average utilization. An HPC application tends to wring out intense computations with mean utilization going as high as 90%. Enterprise contracts and using reserved instances can reduce the high cost of the cloud, but this necessitates meticulous planning and can undermine the pay-as-you-go elasticity offered by the cloud.

It is not uncommon for HPC applications to process or output large volumes of data. Even though cloud object storage is reasonably priced, storage subscriptions that support shared network file system or parallel file systems tend to be expensive.

4.1 Adoptions

The migration of HPC applications to the cloud has become more popular over the years, and some have been published as a research. A case study of the migration of HPC to the cloud will be discussed in this section. Balis et al. (2016) [17] conducted a study to port HPC applications to the cloud, in which they found that using the cloud had a faster "time-to-science". Balis et al. argued that in HPC for research, the turnaround time for scientific projects, "time-to-science", is more important than raw computing power.

Furthermore, companies and organizations have started to migrate their HPC applications to the cloud. HSBC, a banking company hence adhering to stringent regulations, moved 180TB of data and some of their operations to the GCP platform[40]. Another example would be UC Santa Cruz's genomics project using AWS[38], and the claim that their time to process data was cut down from 3 months on their in house cloud to 4 days on AWS. Another notable adoptions would be the Zika Virus Modelling on GCP by Northeastern University[43].

With major cloud platforms expanding their HPC services, running HPC applications in the cloud seems to be getting more recognition not only for scientific purposes, but also commercially in enterprise settings, such as the examples previously mentioned. Though an article in 2018 [11] mentioned aspects such as - performance, networking, storage being the downfalls that would prevent the adoption of HPC in the cloud. However, this article was released before cloud providers started to expand on their HPC services (2019).

A market research conducted by Hyperion Research showed that in 2019[13, 14], 20% of the HPC community run their workload on some kind of cloud platform and that the global market for cloud HPC will reach 6 billion USD by 2023. Though the 2019 research by Hyperion might have an inherent bias as it was sponsored by major cloud providers and this research being cited as a source by white papers released by the cloud providers themselves.

5 Conclusion

This paper conducted a survey of HPC in public cloud elaborating on the challenges and the industry responses for the same. Technologies like High-performance Data Analytics(HDPA) and AI have been the driving factors for new HPC public tools, especially in 2019. Cloud adoption for industries in Genomics, Finance, Water Simulation is on the rise despite in-house clusters being more cost effective and performant than the cloud providers for communication intensive applications. The adoption can be attributed to easier manageability and elasticity provided by the cloud. Various research opportunities exist as the newer tools claiming higher performance have accompanying white papers, blog posts but are yet to be fully vetted by academia. The majority of the peer reviewed papers work on AWS and Azure due to their popularity but comparison of other industry providers like GCP, Oracle, Alibaba provides more research opportunities. A competitive market can lead to a standardisation of APIs to promote inter-operability among clouds.

The answer to the research question depends on the application being used and is still inconclusive due to constantly evolving nature of the public cloud. For various companies and smaller research groups, public cloud is an attractive option as it frees up manpower to research on their areas without employing a major technological team to handle scale. Supercomputers still provide better computational power than the cloud especially for tightly coupled scientific application and both should mutually coexist in the upcoming years. A wise approach for HPC users would be to maximise the use of on-premise infrastructure while making use of the cloud for supplementary computations[19].

The future of HPC seems promising with higher investments by major cloud providers to further reduce the gap between private and public cloud.

References

[1] 30% Of Servers Are Sitting "Comatose" According To Research . https://www.forbes.com/sites/benkepes/2015/06/03/30-of-servers-are-sitting-comatose-according-to-research/.

[2] Accelerate supercomputing in the cloud with cray clusterstor. https://azure.microsoft.com/en-in/blog/supercomputing-in-the-cloud-announcing-three-new-cray-in-azure-offers/.

[3] Aws. https://aws.amazon.com/blogs/aws/aws-data-transfer-prices-reduced/.

[4] Aws vs gcp vs on-premises cpu performance comparison. https://medium.com/infrastructure-adventures/aws-vs-gcp-vs-on-premises-cpu-performance-comparison-1cb3e91f9716.

[5] Edison - cray xc30, intel xeon e5-2695v2 12c 2.4ghz, aries interconnect. https://www.top500.org/system/178443.

[6] GCP Instances. https://cloud.google.com/blog/products/compute/expanding-virtual-machine-types-to-drive-performance-and-efficiency.

[7] Google trends - cloud security 2004-2020. https://trends.google.com/trends/explore?date=all&q=cloud%20security.

[8] HB Series . https://azure.microsoft.com/en-us/blog/hb-series-azure-virtual-machines-achieve-cloud-supercomputing-milestone/.

[9] High performance computing. https://aws.amazon.com/hpc/.

[10] high-performance-computing-in-the-clou . https://www.csc.fi/en/-/high-performance-computing-in-the-cloud.

[11] How the cloud is falling short for hpc. https://www.hpcwire.com/2018/03/15/how-the-cloud-is-falling-short-for-research-computing/.

[12] Once You're in the Cloud, How Expensive Is It to Get Out? https://www.nefiber.com/blog/cloud-egress-charges/.

[13] Smart orchestration speeds hpc workflows in the cloud. https://d1.awsstatic.com/HPC2019/Amazon-HyperionTechSpotlight-Orchestration-Sep2019.pdf.

[14] The importance of expertise for HPC Cloud Computing. https://azure.microsoft.com/en-us/resources/the-importance-of-hpc-expertise-for-hpc-cloud-computing/.

[15] Dean Hildebrand . Competing with supercomputers: HPC in the cloud becomes reality. https://cloud.google.com/blog/products/storage-data-transfer/competing-with-supercomputers-hpc-in-the-cloud-becomes-reality.

[16] Thekkedath. Bala. Challenging the barriers to high performance computing in the cloud. Discussion paper, Amazon Web Services, 2019.

[17] Bartosz Balis, Kamil Figiela, Konrad Jopek, Maciej Malawski, and Maciej Pawlik. Porting hpc applications to the cloud: A multi-frontal solver case study. *Journal of Computational Science*, 18, 01 2016.

[18] Timur Bazhirov. Comparative benchmarking of cloud computing vendors with high performance linpack. pages 1–5, 03 2018.

[19] Mohamed [Ben Belgacem] and Bastien Chopard. A hybrid hpc/cloud distributed infrastructure: Coupling ec2 cloud resources with hpc clusters to run large tightly coupled multiscale applications. *Future Generation Computer Systems*, 42:11 – 21, 2015.

[20] Jaliya Ekanayake and Geoffrey Fox. High performance parallel computing with clouds and cloud technologies. In Dimiter R. Avresky, Michel Diaz, Arndt Bode, Bruno Ciciani, and Eliezer Dekel, editors, *Cloud Computing*, pages 20–38, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[21] J. Emeras, S. Varrette, and P. Bouvry. Amazon elastic compute cloud (ec2) vs. in-house hpc platform: A cost analysis. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 284–293, 2016.

[22] Constantinos Evangelinos and Chris Hill. Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazon's ec2. *Ratio*, 2, 01 2008.

[23] Roberto R. Expósito, Guillermo L. Taboada, Sabela Ramos, Juan Touriño, and Ramón Doallo. Performance analysis of hpc applications in the cloud. *Future Generation Computer Systems*, 29(1):218 – 229, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[24] Abhishek Gupta, Paolo Faraboschi, Filippo Gioachin, Laxmikant Kalé, Richard Kaufmann, Bu Lee, Verdi March, Dejan Milojicic, and Chun Suen. Evaluating and improving the performance and scheduling of hpc applications in cloud. *IEEE Transactions on Cloud Computing*, 08 2014.

[25] Abhishek Gupta, Laxmikant Kale, Dejan Milojicic, Paolo Faraboschi, and Susanne Balle. Hpc-aware vm placement in infrastructure clouds. pages 11–20, 03 2013.

[26] Rashid Hassani, Md Aiatullah, and Peter Luksch. Improving hpc application performance in public cloud. *IERI Procedia*, 10:169 – 176, 2014. International Conference on Future Information Engineering (FIE 2014).

[27] Qiming He, Shujia Zhou, Ben Kobler, Dan Duffy, and Tom McGlynn. Case study for running hpc applications in public clouds. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, page 395–401, New York, NY, USA, 2010. Association for Computing Machinery.

[28] Scott Jeschonek. Azure hpc cache: Reducing latency between azure and on-premises storage. `https://azure.microsoft.com/en-us/blog/azure-hpc-cache-reducing-latency-between-azure-and-on-premises-storage/`.

[29] Hai Jin, Hanfeng Qin, Song Wu, and Xuerong Guo. Ccap: A cache contention-aware virtual machine placement approach for hpc cloud. *International Journal of Parallel Programming*, 43, 06 2013.

[30] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville. Cloud migration: A case study of migrating an enterprise it system to iaas. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 450–457, 2010.

[31] C. Kotas, T. Naughton, and N. Imam. A comparison of amazon web services and microsoft azure cloud platforms for high performance computing. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4, 2018.

[32] Xiaoyi Lu, Jie Zhang, and Dhabaleswar K. Panda. *Building Efficient HPC Cloud with SR-IOV-Enabled InfiniBand: The MVAPICH2 Approach*, pages 115–140. Springer Singapore, Singapore, 2017.

[33] Piyush Mehrotra, Jahed Djomehri, Steve Heistand, Robert Hood, Haoqiang Jin, Arthur Lazanoff, Subhash Saini, and Rupak Biswas. Performance evaluation of amazon ec2 for nasa hpc applications. In *Proceedings of the 3rd Workshop on Scientific Cloud Computing*, ScienceCloud '12, page 41–50, New York, NY, USA, 2012. Association for Computing Machinery.

[34] Overview of Single Root I/O Virtualization (SR-IOV). https://docs.microsoft.com/en-us/windows-hardware/drivers/network/overview-of-single-root-i-o-virtualization–sr-iov-.

[35] A. Prabhakaran and L. J. 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, 2018.

[36] E. Roloff, M. Diener, A. Carissimi, and P. O. A. Navaux. High performance computing in the cloud: Deployment, performance and cost efficiency. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 371–378, Dec 2012.

[37] Iman Sadooghi, Jesus Hernandez Martin, Tonglin Li, Kevin Brandstatter, Yong Zhao, Ketan Maheshwari, Tiago Pais Pitta de Lacerda Ruivo, Steven Timm, Gabriele Garzoglio, and Ioan Raicu. Understanding the performance and potential of cloud computing for scientific applications. *IEEE Transactions on Cloud Computing*, 5(2), 2 2015.

[38] Amazon Web Services. Uc santa cruz genomics institute case study. `https://aws.amazon.com/solutions/case-studies/uc-santa-cruz-genomics-institute/`.

[39] Why using Single Root I/O Virtualization (SR-IOV) can help improve I/O performance and Reduce Costs. https://www.design-reuse.com/articles/32998/single-root-i-o-virtualization.html.

[40] Srinivas Vaddadi. Adopting cloud, with new inventions along the way, charges up hsbc | google cloud blog. `https://cloud.google.com/blog/products/data-analytics/adopting-cloud-with-new-inventions-along-the-way-charges-up-hsbc`.

[41] Edward Walker. Benchmarking amazon ec2 for hig-performance scientific computing. *; login:: the magazine of USENIX & SAGE*, 33(5):18–23, 2008.

[42] J. Zhang, X. Lu, and D. K. Panda. Performance characterization of hypervisor-and container-based virtualization for hpc on sr-iov enabled infiniband clusters. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1777–1784, 2016.

[43] Qian Zhang, Kaiyuan Sun, Matteo Chinazzi, Ana Pastore y Piontti, Natalie E. Dean, Diana Patricia Rojas, Stefano Merler, Dina Mistry, Piero Poletti, Luca Rossi, Margaret Bray, M. Elizabeth Halloran, Ira M. Longini, and Alessandro Vespignani. Spread of zika virus in the americas. *Proceedings of the National Academy of Sciences*, 114(22):E4334–E4343, 2017.

Table 1: Work Distribution

Name	Sections worked on
Abijith	Introduction, Discussion, Conclusion
Gamma	Overview, Comparison - Security, Discussion - Adoption
Tavneet	Comparison - Performance & Cost, Conclusion

The Business Models for Cloud Computing Literature Study

T. van Milligen
Student VU

S.N. Voogd
Student VU

J.M. Kyselica
Student VU

Abstract

Cloud computing has been taking over the way the world buys and uses software. Many, if not most businesses are making the switch to cloud services as opposed to buying traditional software and platforms. In this literature study we take a look at the current state of the business world converting to cloud technology, what business models are being used amongst cloud service providers and what the implications are. We found that there are many potential factors that influence either a company's decision to steer clear of the cloud, or, on the other hand, to embrace new cloud technologies. A combination of multiple business models seems to work best when entering the relatively new cloud market, whereas veterans in the field have the luxury of taking a more specified approach. All in all the topic of doing business with and within the clouds is still quite new and far from an exact science. Future research in this area could prove beneficial for providers and consumers alike.

1 Introduction

Applications and data are disappearing from personal computers and reappearing in 'the computer cloud'. This is a trend that appears to be taking over in almost every domain.
For example, where once you would create a Word document on your own private version of Microsoft Word, you can now log into Google Drive and work on a Google Document, all of which is happening in the cloud, no locally installed software necessary, save for an internet browser. And word processing is just the tip of the iceberg, this type of shift can be found all over the software landscape. Cloud computing is all the rage [1]. This is true for any type of software that can be moved from a privately owned computer to some server operating as part of the cloud, from relatively simple word processing applications like Google Docs, to complete virtual machines. The possibilities have proven to be quite endless.
Naturally, this shift is of great importance for businesses that make use of software and software platforms. Traditional forms of acquiring the technology necessary to run a business is making way for running a business (almost) entirely in the cloud. This is a true paradigm shift and has many implications.

In this paper we will look at the existing studies that have been conducted on how the business world is making the switch to cloud services as opposed to using traditional software and what the business models are behind these cloud services. We will attempt to describe the current state of cloud usage from a business perspective, identify what the current gaps in knowledge are concerning this topic and suggest what future research could be beneficial.

Preprint. Under review.

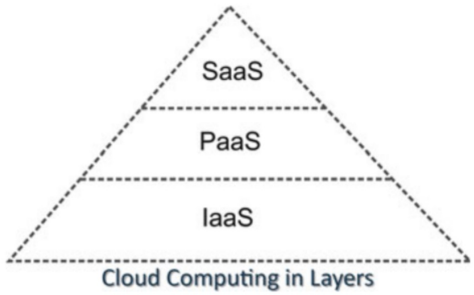


Figure 1: Three three main layers of the Cloud [2]

2 The business models of the Cloud

The cloud can be split up into three main 'layers', each associated with their own service. Figure 1 shows the hierarchy of IaaS, PaaS, and SaaS.

We will now look at these layers in more detail, and discuss their business models.

2.1 Infrastructure as a Service (IaaS) Business Models

Infrastructure as a Service, even just by its name, strongly suggests that it is a kind of utility, comparable to, for example, the electricity grid (see grid computing). It is therefore useful to look at existing business models in the realm of utilities and assess their applicability to IaaS.

Basic infrastructure utilities such as electricity and water use a *'metered usage of service'* business model, as defined by Rappa in [3]. This means that the consumer utilises exactly as much of the resource as they need, and pays accordingly, determined by a predefined fee per unit of the resource (e.g. cents per liter of water). However, other types of utilities, such as telephony and internet access, utilise different business models. Internet access uses a *subscription model for unlimited service*, where the consumer pays a recurring fixed fee, for example each month, to have unlimited access to the service at no additional cost. Telephony uses a business model akin to a kind of hybrid between the two previously mentioned models. It consists of a *subscription model* that grants consumers access to the infrastructure, combined with a *metered usage of service* business model that charges the consumer based on their actual usage of the infrastructure.

In order to link the aforementioned business models to IaaS, it is good to analyse why these different utilities use different business models. Internet service providers and electricity or water providers provide fundamentally different resources. Where electricity and water are essentially goods that have to be produced and are then 'resold' to consumers, usage of the internet infrastructure incurs almost no additional cost to the providers. It is easy to see how the subscription model with unlimited service would be highly unsuitable for electricity and water.

If we now take the IaaS paradigm, we see that it appears to be a mixture between the two. There is a large initial investment for the infrastructure, akin to internet service providers, and providing the service also incurs running costs (in the form of electricity).

Several papers agree that IaaS services should follow the utility model [4, 5, 6]. Furthermore, it appears that the *metered usage of service* (also known as *pay as you go*) model is the best fit for IaaS providers [2]. This can be corroborated by the fact that most of the big IaaS providers (e.g. Amazon AWS, Microsoft Azure) are using the *pay as you go* model to monetize their services. However, according to [2], the *pay as you go* model could be made more flexible and lucrative by combining it with a subscription model.

2.2 Platform as a Service (PaaS) Business Models

The Platform as a Service paradigm aims to simplify the application development and deployment process by providing and managing all the necessary infrastructure for development. This spares the

developers from having to deal with the complexity of maintaining and operating the infrastructure, freeing up more time for the actual development of the applications.

PaaS can take on three different forms, according to the needs of the developers [7]. The first form is PaaS in the form of a public cloud. This means that all customers share an environment which is hosted by the PaaS provider. This gives developers less control, but also increases simplicity and convenience. Some developers might desire more privacy and control over their PaaS, which is when the private form is used. This can be deployed onto on-premise data centers for full control, or onto independent IaaS cloud providers, guaranteeing a secured environment for a single business. Lastly, the two aforementioned concepts can be combined into a hybrid PaaS service, consisting of a mix of dedicated and public PaaS.

Platform as a Service is a more abstract concept than Infrastructure as a Service, which makes it difficult to directly relate to long-established utilities, as was done in the previous section. If we look at one of the first public PaaS services, Zimki, we see that it used a pure *pay as you go* business model. Nowadays, a core component of the PaaS business model is *lock-in*. If an IaaS provider also offers convenient PaaS services that are proprietary to their infrastructure, it is easy to see how the two can cooperate to 'lock developers in' to their services, and increase revenue [8].

2.3 Software as a Service (SaaS) Business Models

Software as a Service resides entirely in the application domain. The developers do not need to concern themselves with the lower layers of the cloud, and can make use of the services from the two lower layers. The main goal of SaaS is to serve software to customers in the form of online services, rather than in the form of local applications that are managed on site. As the concept of SaaS is quite different from IaaS and PaaS, the business models used in Software as a Service also differ.

If we compare SaaS to the utilities mentioned in the IaaS section, we see that it most closely resembles the service of internet service providers. There is a large initial investment to develop the software, but more use of the software does not greatly increase the cost to the service providers. In fact, by building on the lower layers of the cloud, SaaS can be entirely flexible in its running cost. If there are no users using the software, the running cost for the providers will be next to none, as the service will be hosted using a *pay as you go* business model. The more users use the software, the higher the running costs will be. Therefore, the business model has to be chosen in a way that makes this profitable.

The de facto standard solution for the monetization of Software as a Service is the subscription model [2]. This could consist of monthly payments, yearly payments, or other recurring payment schemes. This fits well into the aforementioned use of Iaas and Paas to support the SaaS application. More users means a higher running cost, but with a recurring payment scheme, more users also equals more revenue. This makes SaaS highly flexible.

The subscription model is, however, not the only business model used in SaaS. There are many 'free' SaaS applications that rely on other revenue streams (e.g. Google Docs). However, many of these 'free' applications also serve as a front to the actual subscription based paid SaaS [9].

3 Converting to Cloud

3.1 Risks and Concerns

Ever since the emergence of Cloud Computing, there have been concerns about the risk of switching to a cloud supply chain within businesses. While the benefits of Cloud Computing have become clear, the switch to a new business model impacts the whole organization, and brings with it certain risks. There has been a lot of research done on how to properly manage these risks when switching to a Cloud Computing business model. The article 'Risk perception and risk management in cloud computing: Results from a case study of Swiss companies' [10] shows some of the main concerns businesses had at the time about cloud computing. The article argues that concerns about cloud computing were widely spread around the time of publication, as it cites a survey by the Information Systems Audit and Control Association from 2010 saying that 45% of the surveyed US businesses considered the risks of cloud computing as outweighing the benefits. Some of the main topics of concern include: Regulatory compliance and data location, Availability and disaster recovery,

Information security, and Provider lock in and long term viability. These risks can be classified as political/legal, operational and technical.
The solutions to these problems that the article suggests mostly come down to comparing the companies' needs to the willingness of the provider to fulfill them, and to set up contracts/SLA's according to the companies' technical requirements. The authors conclude that the companies have a good awareness of the risks that come with cloud computing adoption, but with a sample size of only five companies, the relevance of this can be questioned.
Other than that it does provide good pointers for risks to consider and advise on how to handle them for companies from any country that considers switching to cloud computing.

Security risk is something that often shows up as an adoption barrier for cloud computing. In 'Analyzing the operation of cloud supply chain: adoption barriers and business model' [11] the authors cite two surveys [12, 13] in which security is seen as one of the biggest barriers for cloud computing adoption within organizations.
In contrast to this view however, John Nye [14] argues that the specialization offered by security architects at big cloud providers can in fact drive security closer to perfection. This is however not a reason to blindly trust cloud providers, he argues that an extensive risk assessment should always be done when looking to use a cloud provider's services in terms of setting up possible contracts etc., in a similar fashion to how (Brender, Nathalie,Iliya Markov)[10] advised in their study.

In similar fashion, Vemula and Zsifkovits [15] point out that security and privacy are among the main reasons for SMEs to switch to the cloud, while also stating that many others do not consider the cloud reliable. The larger enterprises have other driving factors for switching to cloud, here things like speed of deployment and faster return on investment are driving factors [16].
The increased agility to respond to business requirements with cloud computing in supply chain management and the resulting increase of efficiency is one of the reasons the article recommends cloud supply chain solutions for enterprises looking to improve on supply chain efficiency. More or less the same risks of cloud adoption are recognized as by (Brender, Nathalie,Iliya Markov)[10].

Raut et al. [17] also provide a set of barriers to cloud adoption. The study presents a graph that represents how different barriers interconnect and relate, and which barriers are most significant. According to the authors, their graph based modeling methodology converts unclear, poorly articulated interpretive models into visible, correctly defined models. The findings from their model are that the three most influential barriers to the adoption of cloud computing in business are: lack of confidentiality, lack of top management support, and lack of sharing and collaboration. The adoption barrier of *lack of confidentiality* matches the conclusions of the previously discussed studies very well.

3.2 Managing the transition

How to successfully change the actual business model of a company to fit the cloud computing model from a management perspective is another highly discussed topic.
The article 'Managing potentially disruptive innovations in software companies: Transforming from On-premises to the On-demand (2015)' [18] discusses successful management strategies when switching to the cloud.. The article relies heavily on Christensen's disruptive innovation theory on managing potentially disruptive innovations and the management strategies it offers for dealing with the change [19]. It aims to test whether these strategies also work for the software industry by doing a case study on five companies that went from purely offering an on-premises product to also offering on-demand software products. This study found that Christensen's spin-off strategy was also applicable to these companies. Although the implementation of this spin-off was not always the same, due to companies of smaller sizes not being able to, for example, do a completely disconnected spin-off due to organizational/financial reasons, a separation of resources and responsibility always made the start of the transition easier.
The leader strategy along with offering both types of software in the early stage was a successful strategy for these companies. Joining a new market at an early stage or finding new emerging markets to go along with the new technology worked well for the companies in the study.
The five companies also used expert opinions from things like consultants or cooperation partners and early adopters and said to have benefited greatly from this. The trial and error strategy through learning and prototyping cost a lot of time for all five companies. Two companies learned to avoid

Table 1: Nine management strategies

No.	Strategy	Explanation
1	Spin-off	An independent spin-off, or a separate organizational unit, could help prevent resource allocation conflicts and allow the company to more easily follow potentially disruptive innovations
2	Leader	Preparing the company early and stepping into the market as a leader could be a wise strategy
3	Expert opinion	Gathering information from a wide range of sources (technological staff, cooperation partners, customers, and external experts) and sticking to the adopted path despite resistance (e.g. from shareholders) seems to be a promising strategy to support the transformation process
4	Trial and Error	Test products and test markets could be an important step towards achieving fully developed software. This is especially recommended if the intention is to roll out high quality products (robustness, stability, etc.) in the B2B market
5	Recruitment	Recruiting innovative and experienced staff could help the transformation process. Ideas and innovation may also emerge from cooperation with universities or lead customers
6	Direct sales	It might be best to distribute On-demand software directly. As an alternative, companies could initially financially incentivize resellers to promote On-demand sales
7	Step-by-step	The transformation might be best organized as a step-by-step approach focusing on smaller software solutions in the beginning. Over the course of time, smaller On-demand solutions could expand along with their customer base and thus gain the attention of larger clients
8	Partnership and ecosystem	Committing to a strong technological partner could help companies to adapt to a disruptive technology faster, as this allows companies to gain access to technology and expertise
9	Visionary top management	Inspiring top management can accelerate a transformation and is important to motivate employees

transferring an unchanged On-premised product to the cloud (cramming). The paper claims this might be one of the most important strategies.
Aside from these four management strategies, the case study companies also developed some individual strategies to guide the transformation process. All nine management strategies can be seen in Table 3.

The paper does address some possible dependencies between strategies, as can be seen in Table 2, and dependencies and influences due to other contextual factors.
Furthermore it does a good job arguing for the research method used through citing studies backing it up, and linking the findings to earlier works by, among others, Adner [20] and Christensen [19]. A shortcoming of this paper is that the paper relies for a big part on the recollection of employees at the case study companies, since they were interviewed after the fact.

Having discussed the factors that may be preventing businesses from adopting the cloud in the previous section, we can look at potential solutions that can mitigate these barriers. A 2015 paper called “Investment evaluation of cloud computing in the European business sector” proposes to do this. The study agrees with the barrier to adoption mentioned by [17] of lack of management support, and labels this the most important barrier to the widespread adoption of cloud computing in Europe.

Table 2: Possible dependencies and influences across strategies

Strategies	Possible reasons	Contextual factors
Ecosystem Strategy, Expert opinion strategy	New ideas are generated and discussed in broader context	Independent of a company's size or age
Spin-off strategy, Trial and error strategy	Companies without an independent unit faced harder times integrating test products or markets	Contingent on a company's size
Leader strategy, Visionary top-management strategy	Managers of older companies are more alert to potentially disruptive change and step earlier into a new market. Managers who are willing to take risks also pursue early market entry	Contingent on a company's age and a manager's personal characteristic

Table 3: Critical success-related business model characteristics in the analysis

No.	Critical success-related BMC of the analysis results in the components of a business model
1	Business strategy: know-how transfer, vertical diversification, market expansion
2	Partner network: partners in similar field
3	Resources and activities: know-how –, human –, hardware –, network resource, data/content, production activities, consulting activities, integration activities, comparison and categorization
4	Costs: fixed operational costs
5	Value Proposition: manifold width, -depth, computing service, development environment, -tool, consolidation, cost savings, administration, private –, hybrid cloud, database –, consulting –, integration –, billing –, search –, messaging service, individual support
6	Distribution and customer relationship: print media communication, monitoring, customer community, support, on-site interaction
7	Revenue: one-time-charge, pay-per-use revenue, revenue with supplementary service, membership fees for partners
8	Target market: SME customers, branch market

Their solution to lower this barrier is to come up with a clear cost-benefit analysis method to enable management of companies to evaluate investment in cloud technologies.
Like many of the studies in this field, this study again uses literature study and surveys as the primary source of information. The authors present a very clear-cut 7-step methodology to perform a cost-benefit analysis in the context of adopting cloud technology, which details the exact considerations that should be made, and how they should be made. This could certainly be quite useful to businesses, but the scientific value of this methodology is questionable, as there is no real proof that the methodology really is effective in helping businesses perform CBA, and, as has often proven to be the case, 'expert opinion' is the main source.

3.3 Success in Converting to the Cloud

In an article from 2016 named 'Successful Business Model Types of Cloud Providers' [21], the authors describe their results after analysing cloud providers business models in order to find out which characteristics are related to the success of a cloud business model. Table 3 lists the business model characteristics that were found to be most closely related to success.

The paper also talks about there being three business model meta types:

1. New players offering a cluster of services
2. Experienced players with standardized services
3. Specified providers at high trust level with hybrid solutions and integration support

The authors found that the latter was the most successful, while the first was found to be

the least successful, as it can be difficult for newcomers to compete in the existing cloud market. Naturally, the second type lies somewhere in between.

A paper from 2017 named 'How to Succeed with Cloud Services' [22] attempts to answer a similar question as the previous paper, namely 'how can cloud providers be successful?', by using data from a survey of 596 actual users of cloud services. They mention that the cloud provider market has to contend with severe competition and that customers are not willing to commit to a provider before properly testing out the service they provide. Many businesses start with a 'freemium' business model (as described in the IaaS section of chapter 2), yet fail to generate adequate revenue streams. Other businesses start with a subscription based business model and fail to reach sufficient growth.
The main findings are that it is not sufficient to focus on one element of success or mechanism when studying digital consumer services, which is what the authors claim most previous work has done. In terms of specifics regarding the question 'How to succeed with cloud services", the authors recommend three generic strategies to be applied in practice: development, retention, and habituation. Development refers to the transformation of free users into premium users by managing constraints and dedication. Retention on the other hand, describes a business switching to a subscription model at a certain point in time, while trying to retain as much of their customer base as possible. Habituation is a combination of the two aforementioned strategies. The conclusions of this study do have limited general applicability though, as the survey was only conducted in the context of cloud storage services. However, the paper contains good information and analysis on the ways a cloud business can create revenue, especially in the context of the 'freemium' business model.

For companies looking to make the switch to a cloud based business, it seems useful to look at the factors which have contributed to other businesses' decision to make that switch.
A 2017 paper called "The Importance of Business Model Factors for Cloud Computing Adoption: Role of Previous Experiences" [23] aims to identify the individual business model factors with the highest impact on cloud computing adoption. It does this, similarly to the previously discussed studies, by literature review and interviews with cloud providers and cloud users.
The proposed business model factors can be summarised with: value proposition, provider's capability for cloud computing, customer relationship management, and revenue model and costs. The authors split these factors into more detailed sub-factors. While the authors do seem to provide a good list of potential factors, these are all taken from previous work in their literature review. The main new findings of the study are, in fact, that there is no statistically significant impact of the analysed business model factors on cloud computing adoption. This almost certainly does not imply that the given factors truly have no influence on cloud adoption; more likely it indicates a flaw with the methodology. Indeed, the sample size of the surveys is only 80, so the results of this study have limited use.

3.4 Experiences within the Field

An article from 2015 named 'Cloud accounting: a new business model in a challenging context' [24] focuses on the change in business model the development of cloud services has brought specifically to the world of accounting. Like in many other domains, traditional accounting software was usually bought for personal use and run on the buyer's own computer(s), but with the arrival of accounting software on the cloud, users simply pay a specialized service provider to use the software via the internet.
This article talks about a number of benefits that cloud accounting has compared to traditional accounting software:

- Lowered costs. Buying software requires an initial investment and maintenance costs to keep it up to date. Cloud accounting requires no additional hardware or software licenses. Users can usually pay per use or some sort of subscription fee on a regular basis. In this way, making use of accounting software in the cloud is cheaper than the traditional approach.

- Increased productivity. Cloud services are always available, as long as there is an internet connection. This allows users to make use of their accounting software outside of business hours. Also, location is no longer an issue, as long as the user has a device with them that can connect to the web. The tedious task of backing up data is no longer an issue as this is

offloaded to the cloud provider, leaving the user with more time to focus on other things. Scaling is also made easy with cloud software, as no decisions have to be made beforehand about, for example, how many copies of the software the company will need.

Concerns and risks are mentioned as well. Business owners worry about whether their financial data is safe in the cloud. This is mostly a human concern, as modern cloud services have proven to be extremely secure. Another concern business owners may have is that of losing their internet connection. This however, also proves to be a nonissue as cloud service providers have very strict service-level agreements to ensure that their users do not experience any down-time.
It seems that at the time of writing accountants were still sceptical about the use of cloud software within their field. The authors of the paper take the stance that the accounting profession should indeed adopt cloud technology in order to stay up to date with modern times.

Although it is commendable that the authors attempted to narrow their scope by looking specifically at cloud computing within the context of accounting, it would seem that their conclusions apply to all of cloud computing. The shift from personally owned software to software as a service can be found in all fields.
The benefits and concerns mentioned also apply to the vast majority of domains that are making the shift to cloud computing. The actual work that an accountant does for a company is, of course, different to that of, say, a graphic designer, but as it turns out, the cloud is largely field agnostic, i.e. as long as the necessary software to do one's job is available within the cloud, the fact that the software is found on the cloud has little to no implications regarding the work that is being done.
Financial data is of great importance to a company and should be safe on the cloud, but if it is a key concern of cloud technology to keep all types of data safe, then there really is no point in looking further into the security of financial data specifically.

4 Discussion and Conclusion

In this literature study we have attempted to gain a deeper understanding of how businesses are abandoning their old way of doing things and making the switch to cloud services in order to run their businesses.

First, we have identified what the common business models in each layer of the cloud are, and why this is the case. Next, we investigated what the scientific literature says about making the switch to cloud. We have found that there are many potential factors that influence either a company's decision to steer clear of the cloud, or, on the other hand, to embrace new cloud technologies. Factors such as security, support from upper management, value proposition, long-term viability and regulatory compliance, among others, are high on the list of concerns of companies considering the cloud. We have also found potential methods of mitigating the barriers that prevent businesses from adopting the cloud, such as a clear method of constructing a cost-benefit analysis.

During this literature review we have noticed that every now and then it seems like the business side of cloud computing can become too focused on the hype behind the technology, all the while it does not appear to be entirely sure of what cloud computing is, or what it is capable of. A paper such as 'Cloud accounting: a new business model in a challenging context' calls cloud computing 'innovative' and 'revolutionary', and claims that it can truly change the way accounting is done. However, the reason for this is not adequately explained, which leaves the reader wondering as to what makes doing your paperwork in the cloud so revolutionary and if the author even knows.

Some papers regarding cloud business models appear to do a good job of stating the obvious. Take the paper 'How to Succeed with Cloud Services', published in 2017, for example. Although great thought and work was put into producing the test results, basing them off of a large list of literary sources and even going as far as conducting interviews with experts to back them up, the results themselves do not seem very revelatory. In total 42 business models characteristics are named that contribute to success within the cloud provider industry (see Figure 3), all of which are still extremely broad and non-specific. The conclusion almost may as well have been, 'do everything as well as possible in most areas', which is not very enlightening. The same goes for the business model meta types that are identified. Saying that newcomers with less refined products will have a tougher time surviving than veterans with products fine tuned to their customers' desires is no more than logical. However, as the authors recognize themselves, further research into what makes a successful cloud

business model could result in more specific and useful conclusions, for which this paper could serve as groundwork (which has done a good job of getting the obvious out of the way) and a stepping stone to more interesting findings.

In general, a trend that can be identified in the discussed papers is that there is little 'exact science' to be found. The majority of the papers seem to rely on 'expert opinion', which in and of itself is quite a vague and subjective concept. What we would like to see more of in the field is hard evidence, backed up by experiments, that really shows that what the authors are claiming is really true.

In terms of further research, it seems that there is not very much room for deeper investigation into current business models (e.g. subscription models, pay as you go), as it seems like a fairly straightforward topic, where a sort of 'consensus' is already reached. Perhaps a more interesting path to take is to look deeper into innovative strategies and business models that might be disregarded precisely because such a 'consensus' exists (e.g. IaaS is best suited to pay as you go).

References

[1] Brian Hayes. *Commun. ACM*, 51(7):9 – 11, 2008. ISSN 0001-0782.

[2] Zaigham Mahmood and Richard Hill. *Cloud Computing for Enterprise Architectures*. Springer Publishing Company, Incorporated, 2014. ISBN 144715861X.

[3] MA Rappa. The utility business model and the future of computing services. *IBM Systems Journal*, 43:32 – 42, 02 2004. doi: 10.1147/sj.431.0032.

[4] Artur Andrzejak, Martin Arlitt, and Jerry Rolia. Bounding the resource savings of utility computing models. *HP Laboratories Technical Report HPL-2002-339*, 01 2003.

[5] Tino Vázquez, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. Evaluation of a utility computing model based on the federation of grid infrastructures. In Anne-Marie Kermarrec, Luc Bougé, and Thierry Priol, editors, *Euro-Par 2007 Parallel Processing*, pages 372–381, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74466-5.

[6] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. Eucalyptus : A technical report on an elastic utility computing architecture linking your programs to useful systems, 2008.

[7] Mike Kavis. Top 8 reasons why enterprises are passing on paas, September 2014. https://www.forbes.com/sites/mikekavis/2014/09/15/top-8-reasons-why-enterprises-are-passing-on-paas/7e6d451665aa.

[8] Zachary Flower. Weigh the benefits of paas providers against lock-in risks, May 2018. https://searchcloudcomputing.techtarget.com/feature/Weigh-the-benefits-of-PaaS-providers-against-lock-in-risks.

[9] Sadhana Balaji. Freemium model for saas – the good, the bad, and the in-between, September 2018. https://www.chargebee.com/blog/saas-freemium-model-advantages-and-disadvantages/.

[10] Nathalie Brender and Iliya Markov. Risk perception and risk management in cloud computing: Results from a case study of swiss companies. *International Journal of Information Management*, 33:726–733, 10 2013. doi: 10.1016/j.ijinfomgt.2013.05.004.

[11] Jhih-Hua Jhang-Li and Cheng-Wei Chang. Analyzing the operation of cloud supply chain: adoption barriers and business model. *Electronic Commerce Research*, 17(4):627–660, Dec 2017. ISSN 1572-9362. doi: 10.1007/s10660-016-9238-3. URL `https://doi.org/10.1007/s10660-016-9238-3`.

[12] Wei-Wen Wu, Lawrence W. Lan, and Yu-Ting Lee. Exploring decisive factors affecting an organization's saas adoption: A case study. *Int. J. Inf. Manag.*, 31:556–563, 2011.

[13] Markku Sääksjärvi, Aki Lassila, and Henry Nordström. Evaluating the software as a service business model: From cpu time-sharing to online innovation sharing. In *IADIS international conference e-society*, pages 177–186. Qawra, Malta, 2005.

[14] John Nye. Cloud computing: Are your data secure in the cloud. *ISACA Journal*, 1, January 2015. https://www.isaca.org/resources/isaca-journal/issues/2015/volume-1/cloud-computing-are-your-data-secure-in-the-cloud.

[15] Ram Vemula and Helmut Zsifkovits. Cloud computing im supply chain management. *BHM Berg- und Hüttenmännische Monatshefte*, 161:229–232, 05 2016. doi: 10.1007/s00501-016-0485-3.

[16] Idg: Cloud computing survey 2015, November 2015. http://www.idgenterprise.com/resource/research/2015-cloud-computing-study/.

[17] Rakesh Raut, Pragati Priyadarshinee, Manoj Jha, Bhaskar Gardas, and Sachin Kamble. Modeling the implementation barriers of cloud computing adoption: An interpretive structural modeling. *Benchmarking: An International Journal*, 25:00–00, 10 2018. doi: 10.1108/BIJ-12-2016-0189.

[18] Natalie Kaltenecker, Thomas Hess, and Stefan Huesig. Managing potentially disruptive innovations in software companies: Transforming from on-premises to the on-demand. *The Journal of Strategic Information Systems*, 24:234–250, 09 2015. doi: 10.1016/j.jsis.2015.08.006.

[19] Clayton M. Christensen. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press, Boston, MA, 1997. ISBN 978-1565114159.

[20] Ron Adner. *The Wide Lens: A New Strategy for Innovation*. Penguin UK, 2012. ISBN 9780670921683.

[21] Stine Labes, Nicolai Hanner, and Ruediger Zarnekow. Successful business model types of cloud providers. *Business & Information Systems Engineering*, 59(4):223–233, Aug 2017. ISSN 1867-0202. doi: 10.1007/s12599-016-0455-z. URL `https://doi.org/10.1007/s12599-016-0455-z`.

[22] Manuel Trenz, Jan Huntgeburth, and Daniel Veit. How to succeed with cloud services? a dedication-constraint model of cloud success. *Business Information Systems Engineering*, 61, 09 2017. doi: 10.1007/s12599-017-0494-0.

[23] Kristina Bogataj Habjan and Andreja Pucihar. The importance of business model factors for cloud computing adoption: Role of previous experiences. *Organizacija*, 50, 08 2017. doi: 10.1515/orga-2017-0013.

[24] Otilia Dimitriu and Marian Matei. Cloud accounting: A new business model in a challenging context. *Procedia Economics and Finance*, 32:665–671, 12 2015. doi: 10.1016/S2212-5671(15)01447-1.

5 Worklog

J.M. Kyselica

- Chapter 2
- 3.5 paragraphs (papers analysed) in chapter 3
- Slides in the presentation
- Part of the conclusion
- BibTex reference management

T. van Milligen

- Wrote 3.1
- Wrote half of 3.2
- Slides in the presentation

S.N. Voogd

- Abstract and introduction
- 2 paragraphs (papers analysed) in chapter 3
- Slides in the presentation
- Part of the conclusion

Cloud-based integration - iPaaS

Rico Mossinkoff
UvA ID: 12805157
ricokoff@hotmail.com

Vasileios Ntoumanis
UvA ID: 12623911
vasileios.doumanis@gmail.com

Eric Veliyulin
UvA ID: 13040456
ericvel96@gmail.com

Abstract

As the field of cloud computing is growing, the demand for cloud services is increasing by the day. Users want to have access to all different kinds of web services without too much trouble. This poses a challenge for providers to integrate all these services so the users can combine them. iPaaS (Integration Platform as a Service) is a concept that was designed to help solving this challenge. To determine whether this concept is future proof, we listed the benefits and challenges that this concept poses in practice. We found that there are various benefits, including reduced integration cost, faster time to value, high efficiency and security and improved scalability. We also found that there are still challenges like the combination of overlapping platforms, resource-related and logistical challenges, and integrity issues. When looking more closely at the impact of both the benefits and the challenges, we conclude that iPaaS is here to stay in the future. We think that the benefits outweigh the drawbacks of the concept and future platform will slowly convert to iPaaS infrastructures. However, future research should be done to determine the exact impact of the challenges and if there are more factors that we did not account for.

1 Introduction

Cloud computing has grown a lot in the past years, and the demand of cloud services is higher than ever. With new technologies developing, users want to have access to the latest cloud services without too much trouble. To realize this, providers must have a system to provide these services and to allocate their resources accordingly. This has been a challenge for several years, and it will even become a bigger challenge in the future as the demand is still growing. People often use multiple services at the same time to realize their business goals. The providing systems must be able to integrate these services while maintaining an easy-to-use environment. As a reaction to these challenges, iPaaS (Integration Platform as a Service) has been introduced. iPaaS is a recent development that offers a variety of features to address the problem of using multiple services via one platform. As can be seen in figure 1, many companies are already considering switching to iPaaS.

While this sounds promising, one needs to think about the implications for the future. As every other technology, iPaaS has both advantages and disadvantages. To determine whether this development is here to stay or if it needs to evolve into something better, we need to evaluate iPaaS with respect to the future.
In this paper we will take a closer look at iPaaS and its features and determine if it will still be liable in the future. First we will look at the architecture for iPaaS and the features it holds. We will also give some real-life examples of companies that use this infrastructure. Then we will list some of the advantages that iPaaS offers and try to put them in a future perspective. We will do the same for the disadvantages. Finally we will conclude whether iPaaS is future proof or if has too many flaws.

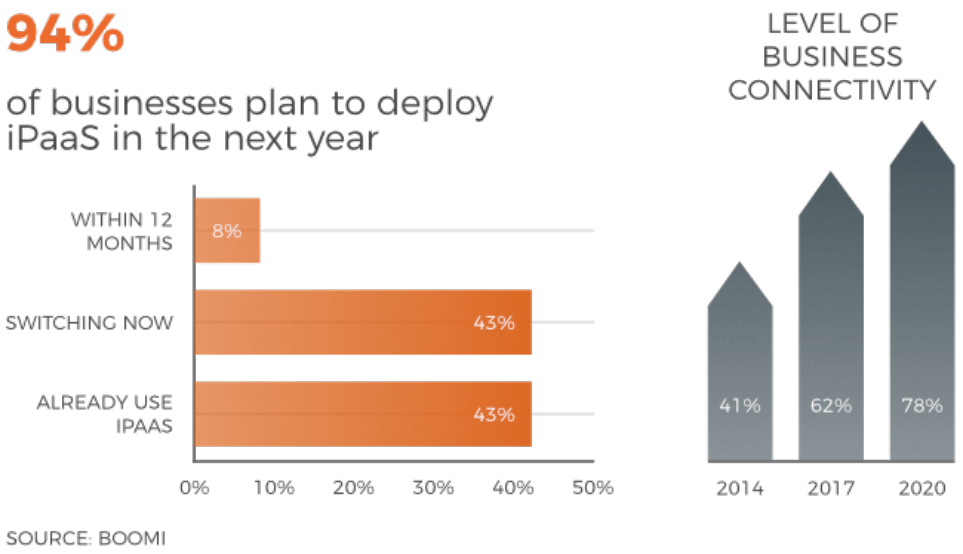


Figure 1: Adoption of iPaaS by companies. Retrieved from https://blog.adverity.com/marketing-ipaas-platforms-future-data-integration

1.1 What is iPaaS?

Integration Platform as a Service (iPaaS) is a cloud based integration system whose purpose is to assist with application, data and process integration, and Service Oriented Architecture (SOA) projects regarding their development and execution platform [17]. iPaaS' competence lies in the field of development, execution, management and govern integration flows.

According to Gartner [6], there are three categories of iPaaS vendors, each focusing on a different area of integration: e-commerce and B2B integration, cloud integration, and Enterprise Service Bus (ESB) and Service Oriented Architecture (SOA) infrastructure. We will shortly discuss these categories to understand their meaning.

1.1.1 E-commerce and B2B integration

E-commerce integration provides the service of exchanging data between a company's e-commerce site and back-end accounting and inventory (ERP) system [2]. As data needs to be synchronized on both sides, integration is key to an efficient system.
According to IBM [1], 'business-to-business (B2B) integration is the automation of business processes and communication between two or more organizations. In this category, the vendor offers a platform to businesses to make it easier to trade and work together with other suppliers and business partners. As most data exchanges can be automated, the platform can take away much of the trivial communication and tasks that are needed between businesses. B2B integration also makes sure that businesses that are using different systems and application can still communicate in a universal way. Maintaining their business network in a fast and agile way is secured in this way.

1.1.2 Cloud integration

With cloud integration, we mean connecting applications, systems and other IT environments to exchange data and services. This kind of integration is widely used over the internet. Almost all enterprises are using cloud computing and integration in some way. It offers a way to connect customers to services and it makes it much easier to scale the business if needed. Because the use of Software as a Service (SaaS) has grown a lot in the past decade, cloud integration has become a vital part in the business strategy of modern companies. Business processes can be optimized and there is a uniform way to share data among different applications, with the use of cloud integration.

1.1.3 ESB and SOA infrastructure

Service Oriented Architectures (SOA) is very similar to the cloud integrations we just talked about. However, there are some small differences. While both are designed to optimally share resources and data [11], we could view SOA as a way to deliver a business service consisting of smaller services. Cloud integrations often contain multiple business services and integrates them to deliver a chain service.

1.2 Real life examples

To understand how iPaaS is used in practice, we will look at some real life applications of it. With this we can see how companies implement iPaaS and in which way they are similar or different. There are many companies in this area, but we will outline only some of them. There might be other very interesting variations that we do not talk about in this paper.

1.2.1 Microsoft Power Automate

Microsoft Power Automate [14] is a cloud based solution to help managing business processes . It uses triggers and actions to automate common processes. The user can set a certain trigger that should start a process of actions. When triggered, actions will follow in a specific sequence to get work done. On the platform there is a variety of different applications that you could use. For instance, you could post a certain message via Microsoft Teams if some other task is finished in your OneDrive. This makes it easier to create a workflow for different applications at the same time.

1.2.2 Adaptris

Adaptris [19] is a company that offers various iPaaS services. One of them is an integration platform that offers similar functionality to Microsoft Power Automate. They also offer services that use the concept of iPaaS. For instance, they offer a service to receive orders and automate response order messages. They also provide custom built integration solutions for specific needs. These solution follow the concept of iPaaS.

1.2.3 Cloud Elements

Cloud Elements [3] offers various services that are based on iPaaS. One of those services are integrations that are embedded in your product. For companies this can be very interesting as it gives them a way to easily integrate their product with others, without worrying about the technical side of implementing such integrations. Cloud elements uses a different approach than some other companies - instead of integrating APIs directly with some application, Cloud Elements creates a virtual API that then can connect with all these APIs. This offers a great opportunity, as it takes away the hassle of integrating all of the different APIs.

1.2.4 Jitterbit

Jitterbit [12] is also a company that offers an integration platform. They state that, with the platform, you can easily connect your cloud applications and you can reuse and extend trusted applications to come up with new innovative solutions. They combine this with artificial intelligence to make this whole process easier. As a platform, it seems that they offer similar functionalities as the other companies we just discussed. However, they have the added benefit of using artificial intelligence, which in turn can make the processes a lot more efficient.

1.2.5 Other systems

Now there are many other systems available in the market that offer similar functionalities. We only discussed a few from which we thought that they are offering a nice example for iPaaS. As we can see in the companies we just discussed, most companies offer the basic integration platform extended with some functionalities. This is what the core of iPaaS entails. The platform is used as a service to connect other services.

2 iPaaS vs other "as a Service" solutions

There are a multitude of other "as a Service" types of solutions available in the cloud market. In this section we will try to explain and differentiate between the three main "as a Service" concepts, namely PaaS (Platform as a Service), SaaS (Software as a Service) and IaaS (Infrastructure as a Service). Additionally, we will compare each of them to iPaaS and analyze the relation they have to each other.

2.1 PaaS

2.1.1 What is PaaS?

PaaS (Platform as a Service) is a type of cloud computing service that works like an environment that supports building, developing and delivering cloud-based web applications. All servers, storage, networking, hardware, software, provisioning and hosting is managed by the provider, while the developers can focus on managing the applications themselves [5] [18].

2.1.2 How does PaaS relate to iPaaS?

iPaaS provides the necessary tools to allow for the integration of the cloud-based web applications that have been created using PaaS. It also serves as a tool to help integrate the data that the applications require in order to run [18].

2.2 SaaS

2.2.1 What is SaaS?

Cloud application services, or SaaS (Software as a Service), is a commonly used cloud computing service that is owned, delivered and managed remotely by one or more providers. SaaS applications are usually consumed on a pay-for-use basis or as a subscription based service. The majority of SaaS applications are made to run directly in a web browser, which makes it so there is no need to download or install anything on the client side [15] [18].

2.2.2 How does SaaS relate to iPaaS?

iPaaS allows organizations to integrate SaaS application data and automate tasks. All iPaaS applications are normally part of SaaS; in other words, they are individual services hosted completely in the cloud. iPaaS acts as a connector between SaaS services and allows for a seamless flow of data between two or more applications. While SaaS applications fully reside in the cloud, iPaaS allows users to integrate services that are both cloud-based, but also on-premise [15].

2.3 IaaS

2.3.1 What is IaaS?

IaaS (Infrastructure as a Service) is a type of instant cloud computing infrastructure that is provisioned and managed over the internet [21]. It provides access to and monitoring of computers, networking, storage and other services. IaaS allows for scaling up and down by purchasing resources on-demand and as-needed depending on a company's business needs, so that they only have to pay for what they use - this is similar to the other "as a Service" solutions [18].

2.3.2 How does IaaS relate to iPaaS?

In the same way that IaaS provides infrastructure in the form of hardware, computing and storage resources, iPaaS should also offer computing and storage resources, at least to a certain extent. However, this will not be offered to the same degree as what you would expect from a pure IaaS solution. Seeing as IaaS provides a foundation for most other platforms, iPaaS is typically deployed on top of IaaS [20].

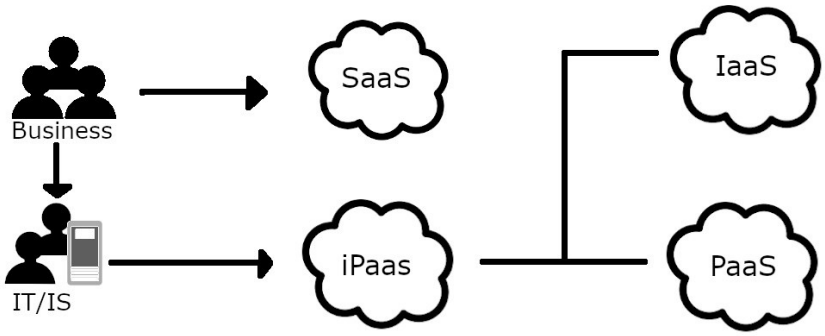


Figure 2: Cloud Services Structure

3 Benefits of iPaaS

iPaaS provides a huge variety of benefits with great ease of use as it comes to technical equipment and infrastructure required to achieve these kind of services. This section will dive deeper into several of these benefits, in an attempt to explain each of them as well as detailing what advantage(s) they can provide to a company's cloud infrastructure.

3.1 Reduced integration costs

Businesses have a variety of costs that they have to make to grow the company and deploy new services. An iPaaS platform automates and facilitates the integration of new services, and as such has the potential to remove much of the costs associated with this. Reduced integration costs could have a big impact in the total costs of a company. This is because the demand for web services is rising and in the coming years, many new services will be deployed. If you can save money on integration of every new service, you can spend that money on other priorities. For a company this means that they will be able to grow faster and they will generate more profit. New companies can also benefit from reduced integration costs, as less initial capital is needed to integrate a new service. The only investment required is an annual subscription for the integration platform. If we look at the near and far future, reduced integration costs can be an important factor to use iPaaS [9].

3.2 Avoided long term costs

Most, if not all, of the companies consider including new services to their environment based on many factors. One of those is the upcoming expenses for deploying these services. In traditional architectures, every service would have quite some extra costs involved, such as costs related to upgrading as well as maintenance costs. This could discourage a company from considering to implement new services. With iPaaS platforms, there is often an option to have a subscription based implementation of it. This means that the extra costs for deploying and integrating a new service will be limited to a fixed price for the use of the platform. This could dramatically decrease costs for companies that often want to deploy new services. Smaller companies that do not have enough resources to maintain and upgrade services can benefit from this solution too. [9].

3.3 Faster time to value

In our times, where DevOps cycles and agile collaborations are key features for companies, conserving time and effort in a development setting is considered mandatory. Every company is trying to adapt to the current state of the cloud market when it comes to collaboration, project management and CRM (Customer Relationship Management), in addition to other business management aspects. iPaaS services provide the best way to remove the workload of preparation and configuration of such an environment within the network of the company and making the services accessible right away. This way, the employees of a company can save time on the usually time-consuming logistical part of deploying cloud services, and instead focus their attention on developing and deploying solutions that actually provide value to their company. [9].

3.4 Efficient, productive and simple

Nowadays a company's development team can benefit from every tool they have at their disposal for every kind of operations they might perform. iPaaS is here to provide a large amount of tools to make employees' life easier from small trivial business related issues, all the way to complex integration flows and scenarios. One of the greatest collections is the pre-built connectors, or integration templates, which are available from the majority of the iPaaS providers. As time passes, a lot more providers are getting involved with more and more visual tools. Some of the most popular tools are the tools of data mapping, transformation and routing. In addition, iPaaS has extended its capabilities to provide adapters/connectors to SaaS (Software as a Service) applications, making the TTM (Time to Market) even better [9].

3.5 Easy to supervise

As large systems are known to be very complex when it comes to management, iPaaS provides the ability to simplify these kinds of operations, such as ability to create, modify and delete a user or a group of users - which is only an example of some the simplest operations it can manage. A bit more advanced activities are available as well, such as assigning user-specific permissions based on their level and even sometimes the ability to specify different permissions for the same user but for different servers. That way, companies can get a clearer and more organized perspective of the users, and are able to manage them more easily [9].

3.6 Improved scalability

Every company has its own needs and these needs can change from one moment to another. One of the most important attributes of an iPaaS platform is the ability to adapt to these requirements and provide the best possible way to handle them. That is why iPaaS includes scalability - this is a flexible way to manage resources either temporarily or permanently. With a great ease of use and without changing plans and increase your cost, companies now have a way to handle any setback and adjust accordingly. For example, they can scale up by adding resources temporarily during specific seasons where it is known or predicted that higher traffic than usual will be occur [9].

3.7 More than just iPaaS

Cloud computing is bringing to the table the way to access our data at any time from everywhere. That alone is a huge benefit for every company that uses cloud computing, as well as for individual employees within these companies, as it allows for much greater flexibility when it comes to for example what devices one can work with, as well as the actual geographical location that an employee can work from. iPaaS provides this constant access to the stored data from everywhere. In addition, all this data can be downloaded, modified and uploaded without stressing the system or making another user wait for some other process to complete first. That is happening because of the load balancing technique that is used. Furthermore, iPaaS provides a disaster plan in events where damages from infrastructure or any malicious software enters the system. Moreover, iPaaS is collaborating with IaaS (Infrastructure as a Service) providers to maintain these services at all times [9].

3.8 Security

One of the biggest concerns from cloud services is security. Preservation of data is important for everyone, from some simple personal files to companies' contracts or other important documents. iPaaS providers reassure the integrity and safety of the data. As the technology evolves and new threats arise every day, iPaaS providers make sure to apply state of the art techniques to prevent any of these attacks that would threaten the security of a company's data. Some of the tools that are used by iPaaS solutions are for example Multi-tenant environment mechanisms, Virtual Private Clouds (VPC), Public Key Infrastructure (PKI), in addition to a host of other tools. [4].

4 Challenges of iPaaS

While iPaaS provides plenty of benefits and opportunities to cloud management, there are also some challenges that must be considered and addressed when implementing this service. Each of the

following subsections addresses one specific challenge, and provides ideas for potential fixes or alternative solutions to avoid these challenges altogether.

4.1 Combination of overlapping platforms

iPaaS platforms are meant to enable and simplify the integration of cloud-based processes, services, applications and data within organizations [7]. However, if a company decides to implement several iPaaS platforms at once, it may add complexity to the enterprise architecture and operational requirements because of the fact that the platforms can have several overlapping capabilities. The cloud and/or on-premise environment of the company can then become unnecessarily complicated, making it more difficult to work with and manage the cloud resources and data [13]. In order to avoid such a scenario, it's important to make sure no iPaaS platforms that are in use are disrupting each other. An even better solution would be to stick to only one iPaaS platform and thus completely bypass any kind of compatibility issues with other platforms.

4.2 Resource-related and logistical challenges

By hosting iPaaS in the cloud, existing cloud-related challenges like sharing of compute resources and cloud outages will also affect the integration service [13]. These potential problems similarly affect all other cloud services, and there is no iPaaS-specific solution to help alleviate this. Another consideration that should be taken when implementing an iPaaS solution is the actual total cost of it - depending on what kind of services and/or the amount of data that the solution has to deal with, running costs can be significantly higher than the initial implementation cost [13]. Platform governance and correct processing of sensitive data is also an important aspect that must be respected when working with iPaaS solutions, so enough IT resources must be delegated in order to achieve and maintain a good level of integrity [22][8]. Before selecting an iPaaS solution to use, companies should evaluate and balance the strengths and weaknesses of the different iPaaS solutions available. This way they can ensure that the selected solution is the one that fits their landscape the best and is the most adapted for their business needs [13].

4.3 Information security and data protection

iPaaS, and other integration platforms in general, often require the need for organizational accounts at external third-party service providers in order to perform external web API calls to these services, for example using REST or SOA. This again requires the issuance of authorization tokens so that the platforms may access and/or consume the services. The authorization process for most integration platforms is normally based on OAuth 2.0, and as such they inherit both the advantages and the drawbacks of this authorization protocol. It is relatively easy to integrate OAuth 2.0 authorization compared to other protocols, as its focus lies on a simplified authorization approach. Because of this, it only provides a limited set of features. As a consequence, the resulting integration lacks fine-grained, policy-enhanced, assured and auditable data flow control and monitoring. Companies are therefore restricted in the degree of control and/or awareness they have over the flow of data and usage of services in distributed environments [10].

4.4 Ensuring data integrity

By using iPaaS platforms' dashboards, developers can easily get information about integrations and potential related errors on a daily basis. If errors are caused by erroneous incoming data, it might be tempting for a developer to fix the error by correcting the erroneous data. This, however, should be avoided at all costs - it is considered bad policy to change incoming data into something else if the failure originates from the sending system. In order to ensure data integrity, iPaaS platforms should be considered only as tools for data transfer and data conversion, and not for editing incoming data content. If corrections have to be made, the developers should inform the owners of the data, i.e. the business system administrators, so that they can make the changes themselves [13].

4.5 Underlying integration issues

In order for a company to fully benefit from an iPaaS platform, it is essential that they understand and address the root causes of the integration challenges that they are currently facing. An example of

such a cause can be data redundancy issues due to immature governance, i.e. unclear data ownership and ineffective problem resolution processes. It is important that governance is addressed upfront, otherwise it can lead to unnecessary integration work. It is also crucial to integrate data quality validation checks throughout the business workflows to ensure long-term data quality - this way, one can avoid scenarios such as the one described in section 4.4. Before data integration projects go live, a thorough clean-up of the data should be performed, in addition to developing a strict plan to maintain its integrity. By first fixing all of the underlying integration challenges within a company, implementing and running a iPaaS platform will prove to be an easier, smoother and more predictable experience [13].

4.6 Discontinuation of iPaaS platforms

Despite the fact that the iPaaS market is showing strong growth, one can already see the first signs of market consoldiation. According to Gartner's prediction, up to two-thirds of existing iPaaS vendors will merge, be acquired or exit the market by 2023. This is because iPaaS vendors face the challenge of having a business that is simply not profitable - the costs for running the platform, in addition to the costs for sales and marketing, outsize the revenue growth and increasing customer acceptance.

Megavendors, for example Oracle, Microsoft and IBM, have the advantage of being able to deliver more-competitive offerings with more-aggressive pricing and packaging options than smaller players in the market, and are as such better-equipped to handle the challenges of the iPaaS market. Gartner expects that this is a trend that will continue, and that it will in turn further diminish the market share of the smaller specialist iPaaS vendors. Gartner argues that this is good news for companies looking to purchase an iPaaS solution - they can capitalize on the evolving market dynamics by solving short-term/immediate problems today, while at the same time preparing to adopt another iPaaS platform from an alternative vendor while waiting for the expected market consolidation to accelerate through 2023.

Market consolidation, however, poses the increased risk of platform services being discontinued because of the vendor exiting the market or being acquired. In order to minimize exposure to this type of vendor-related risk, iPaaS solution purchasers should start by adopting platforms that have the ability to deliver short-term payoffs. This way, the cost of an eventual replacement platform can be more easily justified [16].

5 Conclusion and discussion

iPaaS is a technology that enables and facilitates the integration of a variety of cloud services, processes and data within one or across multiple organizations. There are many benefits to be gained from implementing an iPaaS solution in your company - this includes things such as reduced integration and long term costs, faster time to value, more efficient and productive integration flows, easier management of services and data, improved scalability and security options. There are also certain challenges that one must be aware of when implementing an iPaaS solution - these include risks such as potential underlying integration issues within a company, ensuring data integrity as well as resource-related and logistical challenges. From the point of view of an iPaaS provider, there is also the challenge of maintaining a profitable business, as the costs of running such a platform might be too high compared to their revenue growth.

Most of these challenges, however, are something that can be dealt with if companies dedicate enough time and resources to it. As such, the benefits of implementing and running an iPaaS solution have the potential to outweigh the challenges that are present today. When looking to the future, one can only assume that as companies get more and more familiar with iPaaS as a concept, they will also become more aware of the risks and challenges that must be considered when deploying such a solution. Issues such as potentially overlapping platforms and the lack of data integrity should become less widespread, and companies will be able to truly harness the opportunities that iPaaS solutions provide. There might be factors that this paper did not account for, but based on our research we conclude that companies should consider implementing iPaaS for their integration needs, as this service will only continue to grow in the coming years.

References

[1] *B2B integration*. URL: https://www.ibm.com/supply-chain/b2b-integration. (accessed: 25.05.2020).

[2] Mark Canes. *eCommerce Integration Defined: Understanding Common Software Jargon*. URL: https://www.bluelinkerp.com/blog/2013/02/27/ecommerce-integration-defined-understanding-common-software-jargon/. (accessed: 25.05.2020).

[3] *Cloud Elements*. URL: https://cloud-elements.com/. (accessed: 25.05.2020).

[4] Yuri Demchenko. "Cloud Security services and mechanisms: Can modern clouds provide secure and trusted environment for data and business applications?" In: *Second AMSEC Workshop*. 2019.

[5] IBM Cloud Education. *iPaaS (Integration-Platform-as-a-Service)*. 2019. URL: https://www.ibm.com/cloud/learn/ipaas. (accessed: 26.05.2020).

[6] Massimo Pezzini Eric Thoo Paolo Malinverno. *Gartner Reference Model for Integration PaaS*. 2011. URL: https://www.gartner.com/en/documents/1729256. (accessed: 26.05.2020).

[7] *Gartner Glossary*. URL: https://www.gartner.com/en/information-technology/glossary/information-platform-as-a-service-ipaas. (accessed: 26.05.2020).

[8] Robert Gorwa. "What is platform governance?" In: *Information, Communication & Society* 22.6 (2019), pp. 854–871. DOI: 10.1080/1369118X.2019.1573914. eprint: https://doi.org/10.1080/1369118X.2019.1573914. URL: https://doi.org/10.1080/1369118X.2019.1573914.

[9] *Hybrid IT and iPaaS (integration Platform as a Service)*. URL: http://www.enterpriserealtimeintegration.com/2015/11/12/hybrid-it-and-ipaas/. (accessed: 26.05.2020).

[10] Keith Jeferry et al. "Challenges Emerging from Future Cloud Application Scenarios". In: *Procedia Computer Science* 68 (2015), pp. 227–237. DOI: 10.1016/j.procs.2015.09.238. URL: https://doi.org/10.1016%2Fj.procs.2015.09.238.

[11] Steve Jin. *SOA and Cloud Computing: Are They The Same?* URL: https://blogs.vmware.com/cloudprovider/2010/04/soa-and-cloud-computing-are-they-the-same.html. (accessed: 25.05.2020).

[12] *Jitterbit*. URL: https://www.jitterbit.com/. (accessed: 25.05.2020).

[13] Jonna Metso. "Integration error monitoring in iPaaS environment and implementation model". In: (2019). URL: http://urn.fi/URN:NBN:fi:amk-2019112923428.

[14] *Microsoft Power Automate*. URL: https://flow.microsoft.com/en-us/. (accessed: 25.05.2020).

[15] Shreya Naik. *iPaaS vs SaaS: Know the Difference*. 2018. URL: https://www.built.io/blog/ipaas-vs-saas-know-the-difference. (accessed: 27.05.2020).

[16] Gloria Omale. *Gartner Predicts Up to Two-Thirds of iPaaS Vendors Will Not Survive By 2023*. URL: https://www.gartner.com/en/newsroom/press-releases/2019-03-07-gartner-predicts-up-to-two-thirds-of-ipaas-vendors-wi. (accessed: 26.05.2020).

[17] Vesna Radonjic et al. "Integration Platform-as-a-Service in the Traffic Safety Area". In: Dec. 2011.

[18] Muhammad Raza Stephen Watts. *SaaS vs PaaS vs IaaS: What's The Difference and How To Choose*. 2019. URL: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/. (accessed: 26.05.2020).

[19] *WebSupplier*. URL: https://www.adaptris.com/websupplier/. (accessed: 25.05.2020).

[20] *What is an Integration-Platform-as-a-Service (iPaaS)?* 2019. URL: https://www.dizmo.com/what-is-an-integration-platform-as-a-service/. (accessed: 28.05.2020).

[21] *What is IaaS?* URL: https://azure.microsoft.com/en-us/overview/what-is-iaas/. (accessed: 28.05.2020).

[22] *What Is Platform Governance and Why Is It a Big Deal?* URL: https://learn.g2.com/platform-governance. (accessed: 26.05.2020).

6 Contributions

6.1 Rico Mossinkoff

Responsible for:

- Abstract
- Section 1 - Introduction
- Part of section 1.1 - What is iPaaS?
- Section 1.2 - Real life examples

6.2 Vasilis Ntoumanis

Responsible for:

- Part of section 1.1 - What is iPaaS?
- Section 3 - Benefits of iPaaS

6.3 Eric Veliyulin

Responsible for:

- Section 4 - Challenges of iPaaS
- Section 5 - Conclusion and discussion

MicroVMs and Containers reviewed from a cloud perspective

Marcus van Bergen
University of Amsterdam
marcus.vanbergen@student.uva.nl
10871993

Abstract

With microservices architecture being a solution to build efficient complex software systems, it is important to know which technologies can be used to build such a platform. In this paper we regard Unikernels/MicroVMs and Docker containerization; both can be used to create a microservices architecture. This paper details the underlying technology on which both are built and as to how they are different. Afterwards, we complete this research by elaborating on why Docker containerization seems to be more popular in search trends and research interest. This information provides us insight as to if Unikernels/MicroVMs will become the "next cloud".

1 Introduction

Literature proposes the use of microservices as a solution to efficiently build and manage complex software systems [28]. That same literature lists many benefits of using microservices, such as cost reduction, whilst building complex software. There are a vast amount of software solutions ranging from low-level to high-level, which can be used to build a microservice oriented architecture. In this paper we review two technologies, Unikernels/MicroVMs and Docker containers respectively, which both have been used in ultra-scale platforms performing a service-like role. These services makes part of a larger microservice platform.

In the following sections we will introduce Unikernels/MicroVMs in brief and mention some of their applications. Afterwards, we will detail the underlying technology used to run Unikernels/MicroVMs and discuss the benefits and disadvantages of them. In the sequential section we will perform the same type of analysis but for Docker containers.

Our main research question is: Will Unikernels/MicroVMs become "next cloud"? To answer this research question we will first find answers on the following subquestions: How do Unikernels/MicroVMs and Docker work? Which platform seems to have better security? Why is one more favorable than the other?

These questions will be answered upon in the sections to follow. We conclude this report by giving answer to our main research question.

2 Unikernels/MicroVMs

2.1 A brief introduction to Unikernels and MicroVMs

The recent growth of Cloud computing as a platform to run scalable services has, with it, introduced new technologies. Modern cloud computing allows users to run scalable services which provide users with a flexible platform which can adapt to load. This manner of running services makes

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

cloud providers more responsible for the scalability of services. In the past, the under utilization of cloud resources brought with it great costs. Using virtual machines, which are deployable on an arbitrary hypervisor or hardware, allows cloud providers to better utilize their platforms and lower costs [6].

Unikernels are often small, single spaced, specialized machine images tailored to run a specific application. Common practice for unikernels is to setup them up according to the zero overhead principle: the machine images do not contain superfluous components not required by the application it is deemed to run [9]. Given that unikernels are machine images they are also portable, meaning they can be run on the cloud, a remote server or a personal computer [10]. A typical virtual machine, which has been setup to run one service, is usually based on of a full-fledged operating system which contains many drivers, user spaces, permissions and miscellaneous software such as a text-editor [19]. In contrast, according to the zero overhead principle, unikernels come without many of the aforementioned "overhead" while still being able to run that one service.

2.2 Applications of Unikernels and MicroVMs

The idea of a thin and fast machine image which can run an application seems quite useful. Unikernels have seen themselves been applied in multiple sections of technology. For example, Bromium vSentry, which has now been acquired and renamed to HP Sure Click Enterprise, is based off of Unikernel/MicroVM technology. Bromium's purpose was to start one MicroVM per application on a personal computer. By doing so each "process" is properly isolated from one another and from the host operating system [15]. These MicroVMs run on a hypervisor (Xen in the case of Bromium) which leverage CPU Virtualization Technology; allowing the virtual machines to be hardware accelerated (very fast) and isolated up to hardware level. Isolating each application brings enriched protection to common ransomware attacks or zero-day attacks in general, because the attack-surface for such exploits is limited to that MicroVM. In the case of a ransomware attack, the infected process should not be able to access the hostOS. Such exploits are then only limited to the MicroVM due to their strict isolation.

A different use of MicroVMs is its application in the Amazon Firecracker platform. Firecracker is a Virtual Machine Monitor (VMM) designed for running serverless applications, containers and functions [2]. Firecracker was specifically made to support two AWS (Amazon Web Services) services, Lambda and Fragate. At launch, on the AWS Lambda platform, each Lambda function was run in a Linux Container and these containers ran on a single VM per customer. By doing so, customers' functions were isolated at VM level, but per customer each function was only isolated at container level. The drawback that Amazon realized with this approach, was that fixed-sized VMs per customer lead to underutilization at the cost of security. Instead, they decided to migrate their Lambda platform to Firecracker, where each Lambda function was a single MicroVM instance. The Firecracker platform uses KVM (the Linux Kernerl-based Virtual Machine) as the hypervisor via which they run the MicroVMs. On Firecracker QEMU, which is often used as the virtual machine monitor, via which the MicroVMs are configured, managed and have hardware emulated, was replaced by their own implementation [2]. By running the Lambda functions as MicroVMs, Amazon is now able to achieve multitenancy of Lambda functions on a shared host, while keeping isolation standards high, overhead low all the while increasing economies for their serverless platform.

2.3 MicroVM technology

The aforementioned sections have introduced us to the concept of Unikernels/MicroVMs. In this section we will further discuss the underlying technological stack needed to run MicroVMs.

The Amazon Firecracker example has already hinted what is needed to run MicroVMs successfully. First, virtual machines need hardware, but because they are virtual, the hardware needs to be emulated. Commonly, QEMU which is an opensource hardware emulator, is used for this. In the case of Firecracker, Amazon chose to not use QEMU and instead replaced the hardware emulation with their own implementation. Besides hardware emulation, often hardware-assisted virtualization is used in conjunction in order to allow the virtual machine to make use of the CPU Virtualization Technology. In Firecracker's case they use KVM. This hardware-assisted virtualization allows the emulated hardware to make use of hardware acceleration which achieves speedup, compared to normal software emulation, and allows the emulated hardware to function at near bare-metal speeds.

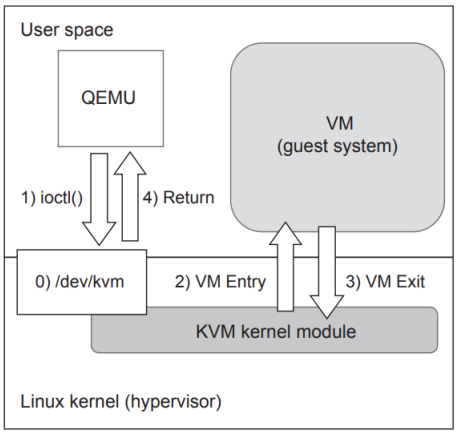


Figure 1: A linux-based host is running a VM (guest), using both QEMU (to emulated hardware) and the KVM kernel module to execute CPU instructions directly on hardware. If a call cannot be directly executed on hardware then the call is passed back to QEMU to be emulated. Courtesy of [14]

It is now clear that the basis to run MicroVMs is a combination of a hardware emulator like QEMU and a hardware-assisted virtualization platform like KVM. With the zero overhead principle in mind, MicroVMs often do not have much emulated hardware attached. We have also noticed that what hardware is attached to the MicroVM is implementation specific. In the case of Amazon Firecracker, only the following emulated devices are supported to attach to a MicroVM: network and block devices, serial ports, and PS/2 keyboard controller [2]. IncludeOS, which is an example of a Unikernel operating system for C++ services, only supports networking and storage [9]. This is an example where we see heterogeneous support for MicroVMs that is platform specific.

2.4 Benefits and disadvantages of MicroVMs

Given the acquired knowledge regarding MicroVMs from the previous sections we will now attempt to understand their benefits and disadvantages. We notice that the literature brings to light observations of MicroVMs which might often times can be both a benefit and a disadvantage. For example, [8] mentions security observations found when using MicroVMs including but not limited to: reduced software attack surface, the use of a single address space, no shell by default.

Reduced software attack surface. [8] argues that due to the zero overhead principle, the number of bytes for MicroVM images are often smaller than normal machine images. In their example, they compare a TinyCore Linux image to that of a MicroVM and find a 92% reduction in image size, and thus a reduction in software attack surface of that same amount. Of course, a smaller machine image comes at a cost that certain software features are missing, which leads us to the next benefit and disadvantage.

No shell by default. As elaborated on in [8], most unikernel systems run without a shell. Commonly, system administrators usually log into a system hosting a service to edit configurations or read the logs. For unikernel systems, where there is no shell, this would not be possible. [8] suggest that other means of logging could be for example, via serial port or over a secured network to a logging system. By withholding a shell from unikernel systems, executing malicious code becomes more difficult for hackers. It is possible to build a unikernel OS where a shell exists, however [8] suggests not including a shell by default for production environments.

Isolation from the hostOS via virtualization. From the aforementioned sections it should now be clear that common practice for running MicroVMs is to use hardware emulation and hardware-assisted virtualization, we will refer to this as the KVM/QEMU stack.

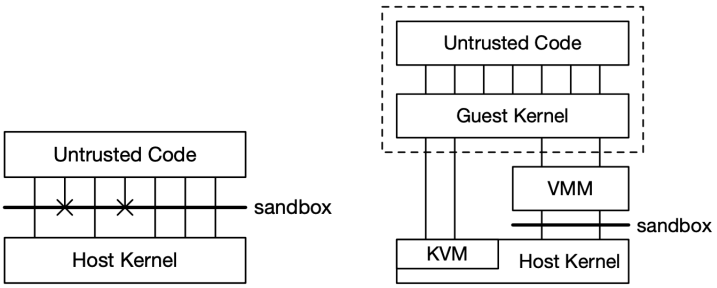


Figure 2: On the left, Linux container model for running applications on a host. On the right a KVM/QEMU model for running applications on a host. Courtesy of [2]

Figure 2 demonstrates how KVM/QEMU stack sandboxing works compared to that of a containerization model. Using the KVM/QEMU stack proper security is not reliant on the host kernel but instead on the VMM (virtual machine manager). This contrasts to the containerization platform where security is reliant on proper implementation of the Linux kernel features such as: cgroups, namespaces etc. Furthermore, the use of KVM allows the virtual machines or the VMM to execute commands on the CPU directly using the CPU virtualization extensions. This contrasts to how containers run, where the host kernel is used to run applications as processes.

Resource footprint. As [9] mentions, a "Hello World" service using IncludeOS uses 8.45 MB in memory (RAM), compared to a Java implementation (no operating system) which uses 29.29 MB in memory. To further enhance this contrast, a full-fledged Ubuntu 14.04 virtual machine running the hello world service uses close to 300 MB in ram. Of course, this enhanced resource footprint is also carried over to the image sizes of MicroVMs and their startup time. Both of these are often lower when compared to normal virtual machines. For example, for the AWS Lambda platform it is important that MicroVMs start almost instantaneously, compared to normal virtual machines which take upwards of seconds to start [2]. The Firecracker VMM + KVM stack is able to achieve this near instant startup-time where a MicroVM is able to start within 125ms.

3 Docker Containers

3.1 A brief introduction to Containerization

Containerization is a ubiquitous technology which has seen widespread use by cloud providers, personal developers and even by ultra scalable platforms. As mentioned in [21] Docker is the most popular containerization solution. We will be limiting the containerization scope of our literature review to Docker containers. It is important to note that other containerization solutions do exists such as LXC (Linux Containers). Containerization in general is not a new technology as its recent uptake in popularity would suggest [20]. Rather, other technologies such as FreeBSD Jails, Solaris Zones all predate Docker [26].

Containerization is an operating system level virtualization. As mentioned by [20], Docker containers try to solve the conflicting dependency problem, missing dependency problem and platform difference problem. Docker attempts to solve the conflicting dependency problem by isolating individual services to their own containers. For example, if one Python service needs Python 3.6, while the other needs Python 3.7, these application run on two separate containers which contain their respective Python versions. Running applications in their own containers further contrasts with the classical method of "one virtual machine for several services". The same goes for the missing dependency problem. If an application needs a certain dependency to run, it is common practice to install this dependency in the image where the application too will be installed. Therefore, when this image in run, in form of a container, the dependency is always satisfied for that application. Finally, the platform difference problem seems to be one of Docker's strongest points. If an application needs to be run on a server with a different OS, simply install Docker. Afterwards you can run the image as a container and all should work [20].

The above information emphasizes why Docker seems so attractive for developers and cloud providers.

4 Applications of Docker

The previous section already hints to some applications of Docker. The wide-spread adoption of Docker has allowed it to gain a large community. With such a large community many projects have been established which make use of Docker containerization or improves it in a certain way.

For example Kubernetes, which is a container orchestration tool that aims to decouple containers from the systems on which they operate. Kubernetes aims to leverage Docker containers to deliver large-scale distributed systems. By decoupling containers from the systems they run on, and weaving them with large-scale robust networks, the application developers can view the containers as a unit of computation [7]. An orchestration tool, such as Kubernetes, then takes care of how many containers need to be available to satisfy load requirements. In this case, by packaging your applications in a Docker image, not only are the problems: conflicting dependency problem and the missing dependency problem solved, but also the system on which the container runs becomes abstracted. Rewriting applications to micro-service form allows organizations to become more productive while allowing applications to become scalable [24]. Furthermore, [18] forecasts that 80% percent of app development on cloud platforms will for micro-services. With the prevalence of cloud platforms and this forecast, the importance of Docker might only further increase.

In [4] it is pointed out that Docker containers are also often used for continuous integration/continuous delivery (CI/CD) applications. [4] argues that if one were to use a traditional VM for CI/CD, it could take upwards of 10 minuets to setup and tear down said VM (with Jenkins, a build tool). Instead, this process could be sped up by orders of magnitude by replacing said VM with a container running Jenkins. Further use of Docker containers in the CI/CD space can be found with their application as GitLab runners [5]. This study shows how one can use Docker containers to validate changes on the git mainline. The techniques mentioned in [5] are a similar implementation of the Jenkins example [4] mentions.

4.1 Docker container technology

Now that we are more familiar with Docker containerization and its applications, we will further our research into the details of this technology. As Figure 2 suggests, there are differences in how containerization and MicroVMs operate. Even though their low level manner of operating is different from virtual machines, containers are sometimes referred to as “lightweight virtual machines” as found in [20].

In contrast to MicroVMs, elaborated on in Section 2, containers do not require emulation of hardware. Instead, [11] explains that in a Linux environment containers often offer isolation and resource management on the host. In fact, all containers on a host share the same host kernel. [11] argues that by not requiring hardware emulation, and by using the host kernel, containers provide near native performance in contrast to full-fledged virtual machines. Furthermore, [20] mentions containers are not aware they share host resources.

In the case of Docker, all Docker containers on a host run on the Docker engine. The Docker engine is the part of the Docker software stack that is responsible for, amongst others, running, scheduling and networking containers [23]. The Docker engine is built in a client-server fashion, where the server is commonly referred to as the Docker engine’s daemon. This daemon exposes a RESTFUL API (application programming interface) which a Docker client can access. A Docker client can thus send commands to the engine to directly manage containers [16].

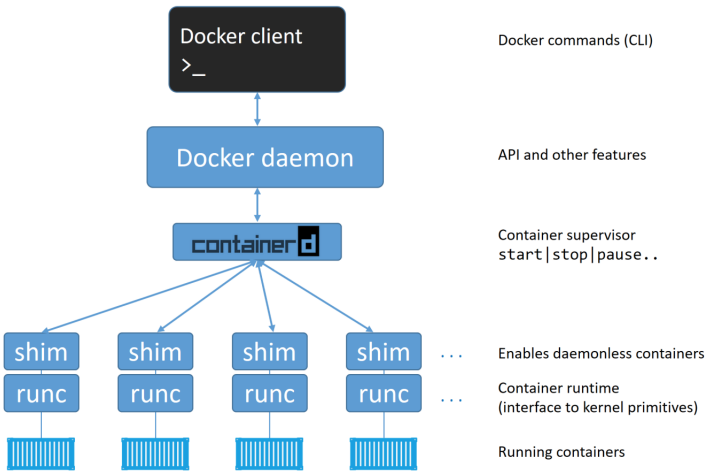


Figure 3: An exploded-view of the components which makeup the Docker Engine and how the client connects to the Docker daemon. Courtesy of [23]

Previously, the Docker engine used to be a monolithic application. Throughout that time, it relied on LXC as the execution environment for containers to run. Since then, the engine has been refactored into what is shown in Figure 3. The benefit of refactoring the Docker engine have been large, but we will highlight two major benefits. First, migrating away from LXC as the default execution environment meant that the Docker engine could be platform-agnostic; instead of being dependent on LXC and the endpoints it exposes to run containers via the host kernel as [23] mentions. Furthermore, as shown in Figure 3, runc is the container runtime implemented to run Docker containers. By implementing runc, Docker pushed themselves to make their container runtime specification and container image specification OCI (Open Container Initiative) compliant [23]. This means that Docker containers follow two open source container-related standards [23]. In theory this would mean that clients using Docker containers should be less prone to vendor lock-in, as the container should be able to be run on other OCI compliant runtimes.

When an OCI compliant image is passed to runc, runc is then in charge for interfacing with the host kernel and afterwards creates the container. During the interfacing process, runc is allocated many things of which the namespace and cgroup are very important. This namespace is an isolated environment where the container's access is limited to. Furthermore, the cgroups or better known as control groups are used to limit the resources a container may use. Both the namespace and control groups help run the containers securely and responsibly on a host.

4.2 Benefits and disadvantages of Docker containers

Given Dockers widespread use it should not be a difficult task to find the benefits of the technology. In this section we will provide some benefits and disadvantages of Docker.

For starters, [29] compares monolithic applications versus microservice patterns. The general consensus that [29] provides is that microservice patterns make it more practical for companies to maintain a large code base. Furthermore, the microservice pattern allows services to be independently scaled and deployed. [17] then continues by elaborating that Docker would be a good fit to implement such a microservice architecture.

[17] further mentions multiple development-friendly components Docker provides such as image registries, where Docker images can be uploaded to (push). A user can then use a Docker client to pull (download) the images to their own Docker server and run the images [17]. [17] also goes further and elaborates on the Dockerfile, which is a simple script that contains a descriptive set of instructions showing how a Docker image is built. Further benefits previously elaborated on were to use Docker containers inplace of VMs to develop a CI/CD pipeline [5]. The container would, in this context, achieve great speedup compared to traditional VMs.

In the previous section we have elaborated on how containers are different than VMs. Research found in [1] demonstrates that containers can be faster than VMs. [1] speculates this to be the case due to a container not having to startup a separate guest OS, as is the case for a VM. This highlights another advantage of Docker containerization, which is that containers share the host resources. With this, comes disadvantages that [13] highlights in their paper. [13] concludes that due to incomplete implementation in system partitioning technology, which is the same technology Docker uses to securely use host resources, it would be possible to jeopardize the system on which the containers run on. Further research performed by [12] concludes that the layer of hypervisor between a VM and a host is often considered thicker (security wise) than the layer between container on host.

Finally, one of the largest benefits of using Docker containers is their portability. It is possible to run Docker containers on a PC by installing the Docker Engine. Aside from this, it is also possible to run these containers on many cloud platforms such as Google Cloud, Amazon Elastic Container Service etc. Furthermore, it is possible to leverage the power of an open source orchestration tool, such as Kubernetes, to deploy a containerized application or service to an ultra scalable platform [1].

5 MicroVMs/Unikernels and Docker containers: Popularity over time

So far, this review has discussed two technologies Unikernels/MicroVMs and Docker containers respectively. As we have explained thus far, the purpose of both technologies is to leverage and underlying technological stack to run a certain application or service. In the case of Unikernels this could be an Amazon Lambda function. For Docker containers it could be one service which makes part out of a large microservice architecture. Recall Figure 2 which gives a high-level overview on the differences in implementation regarding MicroVMs and containers. Till now, we have been able to establish that MicroVMs run on a certain VMM and hardware emulation stack. In contrast, containers directly make use of existing host resources, albeit in a secure manner by using cgroups and namespaces.

Certain literature like [22] suggests that the Unikernels are an answer to the many security issues currently plaguing the cloud atmosphere. Regarding Docker security issues, [27] has found that on average both official and community Docker images contain 180 vulnerabilities. Further research by [27] found that many times vulnerabilities are propagated from parent images to child images. [27] also found that images are not updated frequently enough, whereby vulnerabilities which have been patched are not addressed with an image update. This same research concludes that Docker needs more systemic methods of applying security updates; something it is currently missing.

With this in mind, and even with the large benefits which Unikernels/MicroVMs bring (outlined in Section 2), Docker seems to be as popular as ever.

Figure 4: Google Trends search index for search queries: “unikernel” or “microvm” (blue) and “docker” (red). It is clear that interest in Docker shows a large upward trend, while for unikernels/MicroVMs no such trend is apparent. Interest seems to have stagnated.

As Figure 4 shows, even with the security issues plaguing Docker images, the idea that Unikernels will be an answer to Docker issues seems not to realize. Literature [25], speculates that after

[1] https://cloud.google.com/kubernetes-engine

the initial Docker adoption, Unikernels/MicroVMs will be the "next cloud". Even so, it is apparent from Table 1 that research interest in Docker and containerization seems to dwarf research in Unikernels/MicroVMs.

Search Query	Number of Hits
Unikernel OR MicroVM	220
Docker AND conterinerization	144
Docker	2165
conterinerization	1288

Table 1: Number of articles matching the given search query. Is is apparent that research interest in containerization is much more significant that of Unikernels/MicroVMs. Courtesy of `https://www.scopus.com`

At this point we dare speculate as to what factors have played a role in Docker's success, and why MicroVMs/Unikernels have not become the "next cloud". First, lots of Docker's software is open source and follows a freemium business model. It is very easy to get started with Docker. The installation to get-it-running process is very streamlined. The fact that there is a community edition of the Docker engine allows users to gain experience with the product for free. In the case of MicroVMs/Unikernels it is less clear how precisely one gets started. Also, as literature has stated many of Docker's products are quite user friendly and intuitive such as the Dockerfile [17].

We also believe that one of the large benefits of Docker containers is that they comply with established standards. At this moment of writing we have been unable to find a group which standardizes Unikernel/MicroVM practice. On the contrary, Docker has worked very closely with the Open Container Initiative to standardize operating-system virtualization. As we have explained in Section 3, Docker has adapted their engine to comply with these standards.

The widespread adoption Docker has seen by large companies and its community is also apparent. We notice that many successful products make use of Docker containerization, such as Kubernetes or GitLab CI/CD, which presumably help make Docker containerization more attractive. Furthermore, there are countless cloud providers (Amazon ECS, Microsoft Azure, ...) which offer Docker as a service. On such platforms scaling an application which is packaged in form of a container is managed by the platform itself.

Finally, it is clear that Docker is reactive to the community. As [23] highlights, when Docker noticed that their engine was becoming monolithic, and thus too slow, they decided to refactor and redo its implementation. Making use of runc as the execution environment has made the Docker engine better as containers are now daemonless. Furthermore, a thrid-party audit by Cure53 [3] on runc has deemed the execution environment safe. This however does not guarantee that Docker is as safe as MicroVMs/Unikernels. Even so, these audits possibly makes articles like [22], claiming Unikernels to be the answer to many security issues, less relevant.

6 Discussion and Conclusion

In this paper we have reviewed two technologies, which can be used for the same purpose, but at their core are different in implementation and definition. We have given answer to the subquestion: How do Unikernels/MicroVMs and Docker work?

UniKernels/MicroVMs are based off of true virtualization. We have established that for Unikernels/MicroVMs a hypervisior and hardware emulation is needed in order to run. The benefits of this is that having such a hypervisor adds a "thicker" layer of security when compared to containerization. Furthermore, UniKernels/MicroVMs are not dependent on the host resources to operate. UniKernels/MicroVMs are also often built with the zero principle in mind, where the amount of software and resources distributed to a MicroVM is not more than needed by the application it is running. By doing so, UniKernels/MicroVMs are often low in resource footprint. Furthermore, this atypical philosophy results that Unikernels/MicroVMs often contain a stripped-down version of an OS. Common practice is that such an OS has no userspace or shell included. This commonly results in a smaller software attack-surface and faster boot times making them exceptional for use in ultra-scale platforms such as AWS Lambda.

On the other hand we have also investigated containerization technology, more specifically Docker containerization. In this research we have been able to establish the different methodology which is used to run containers. In contrast to MicroVMs, containers do borrow host resources, relying on proper kernel isolation features such as name spaces and control groups to run containers securely. The Docker engine, and its implementation, have improved over the years. Moreover, Docker software has proven to be user-friendly, audited, platform agnostic and based on open source specifications. Docker has seen widespread adoption and have continued to stay reactive to the community. Furthermore, they contribute open source, open standards and improve their own technology and containerization in general. With its wide spread adoption many projects have come into existence based on Docker containerization, such as Kubernetes.

We have also regarded the subquestion: Which platform seems to have better security? Literature has shown that in general, even though runc has been deemed safe by security audit, Unikernels/MicroVMs do seem more secure than Docker containers. This generally has to do with the fact that Unikernels/MicroVMs have a thicker "layer" of protection than containerization. Other literature has also noted that incomplete, or work in progress implementations of kernel security features for containerization (namespaces and cgroups), might result in jeopardized Docker hosts. Furthermore, literature also suggest that Docker images often have many vulnerabilities and suggests Docker to address this problem with a structural solution.

In Section 5 we given answer to Why one technology, Docker, is more favorable than Unikernels/MicroVMs? It is difficult to answer this concretely, however we do speculate as to why we believe this to be the case. First, Docker has been reactive to the market and has improved its products over time. While being reactive to the market and refactoring the Docker Engine, Docker adopted and helped contribute to the Open Container Initiative. Adopting OCI runtime and image standards presumably makes Docker containers more attractive, as they should theoretically work on other OCI compliant runtimes. In the case of MicroVMs/Unikernels we have yet to find any specification for the machine images or how they should execute. Furthermore, other large technical projects such as Kubernetes or GitLab CI/CD use Docker containerization in their platform. This possibly boosts the confidence in Docker containerization. On the contrary, Amazon Firecracker has also proven that Unikernels/MicroVMs are suitable for production. Literature also notes that Docker products are user friendly and intuitive which obviously makes them appealing to consumers.

From a cloud perspective, both technologies deem fit for ultra scale use as proven by their success in Amazon Firecracker and Google Kubernetes Engine respectively. Even though certain literature promises UniKernels/MicroVMs to be the "next cloud", we have yet to see this be realized. Furthermore, it is apparent that at this time of writing, interest in Docker and containerization seems to be much more present than that in UniKernels/MicroVMs. Given its current rate of maturity, contributions to open source and general (positive) consensus, it would come as no surprise to see Docker continue to trend upwards; which weakens the point that Unikernels/MicroVMs will be the "next cloud". Nonetheless, given that UniKernels/MicroVMs use true virtualization, and Docker uses containerization, it is understandable why certain parties choose the former over the later; as virtualization does bring with it some appealing advantages.

References

[1] Theodora Adufu, Jieun Choi, and Yoonhee Kim. "Is container-based technology a winner for high performance scientific applications?" In: *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE. 2015, pp. 507–510.

[2] Alexandru Agache et al. "Firecracker: Lightweight Virtualization for Serverless Applications". In: *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*. 2020, pp. 419–434.

[3] Ing et. al. *Security Review Report runc*. 2019. URL: `https://github.com/opencontainers/runc/raw/master/docs/Security-Audit.pdf`.

[4] Charles Anderson. "Docker [software engineering]". In: *IEEE Software* 32.3 (2015), pp. 102–c3.

[5] Mohammed Shamsul Arefeen and Michael Schiller. "Continuous Integration Using Gitlab". In: *Undergraduate Research in Natural and Clinical Science and Technology Journal* (2019), pp. 1–6.

[6] Anton Beloglazov and Rajkumar Buyya. “Energy efficient resource management in virtualized cloud data centers”. In: *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE. 2010, pp. 826–831.

[7] David Bernstein. “Containers and cloud: From lxc to docker to kubernetes”. In: *IEEE Cloud Computing* 1.3 (2014), pp. 81–84.

[8] Alfred Bratterud, Andreas Happe, and Robert Anderson Keith Duncan. “Enhancing cloud security and privacy: the Unikernel solution”. In: *Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 19 February 2017-23 February 2017, Athens, Greece*. Curran Associates. 2017.

[9] Alfred Bratterud et al. “IncludeOS: A minimal, resource efficient unikernel for cloud services”. In: *2015 IEEE 7th international conference on cloud computing technology and science (cloudcom)*. IEEE. 2015, pp. 250–257.

[10] Antonio Corradi, Mario Fanelli, and Luca Foschini. “VM consolidation: A real case based on OpenStack Cloud”. In: *Future Generation Computer Systems* 32 (2014), pp. 118–127.

[11] Rajdeep Dua, A Reddy Raja, and Dharmesh Kakadia. “Virtualization vs containerization to support paas”. In: *2014 IEEE International Conference on Cloud Engineering*. IEEE. 2014, pp. 610–614.

[12] Michael Eder. “Hypervisor-vs. container-based virtualization”. In: *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)* 1 (2016).

[13] Xing Gao et al. “ContainerLeaks: Emerging security threats of information leakages in container clouds”. In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2017, pp. 237–248.

[14] Yasunori Goto. “Kernel-based virtual machine technology”. In: *Fujitsu Scientific and Technical Journal* 47.3 (2011), pp. 362–368.

[15] Bromium HP. *Secure Browsing for the Era of the Mobile Worker*. Tech. rep. HP Development Company, L.P., Nov. 2018.

[16] Docker inc. “Introduction to Container Security Understanding the isolation properties of Docker”. In: 2016.

[17] David Jaramillo, Duy V Nguyen, and Robert Smart. “Leveraging microservices architecture by using Docker technology”. In: *SoutheastCon 2016*. IEEE. 2016, pp. 1–5.

[18] Xabier Larrucea et al. “Microservices”. In: *IEEE Software* 35.3 (2018), pp. 96–100.

[19] Anil Madhavapeddy and David J Scott. “Unikernels: the rise of the virtual library operating system”. In: *Communications of the ACM* 57.1 (2014), pp. 61–69.

[20] Dirk Merkel. “Docker: lightweight linux containers for consistent development and deployment”. In: *Linux journal* 2014.239 (2014), p. 2.

[21] Claus Pahl. “Containerization and the paas cloud”. In: *IEEE Cloud Computing* 2.3 (2015), pp. 24–31.

[22] Russell Pavlicek. *Unikernels*. O’Reilly Media, Incorporated, 2016.

[23] Nigel Poulton. *Docker Deep Dive*. Amazon, 2017.

[24] Harika Rajavaram, Vineet Rajula, and B Thangaraju. “Automation of Microservices Application Deployment Made Easy By Rundeck and Kubernetes”. In: *2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE. 2019, pp. 1–3.

[25] Darren Rush. *After Docker: Unikernels and Immutable Infrastructure*. 2014. URL: `https://articles.microservices.com/after-docker-unikernels-and-immutable-infrastructure-93d5a91c849e`.

[26] Thorsten Scherf. *Application virtualization with Docker*. 2015. URL: `https://www.admin-magazine.com/Archive/2015/29/Application-virtualization-with-Docker`.

[27] Rui Shu, Xiaohui Gu, and William Enck. “A study of security vulnerabilities on docker hub”. In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 2017, pp. 269–280.

[28] Andy Singleton. “The economics of microservices”. In: *IEEE Cloud Computing* 3.5 (2016), pp. 16–20.

[29] Mario Villamizar et al. “Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud”. In: *2015 10th Computing Colombian Conference (10CCC)*. IEEE. 2015, pp. 583–590.

Literature Review Participation

I found myself in a peculiar situation. Previously I was member of group 14 which had trouble working together, we have since split. The details are known to the faculty in charge of this course and the TAs grading.

I found it uncomfortable working with my group given the copy-paste I discovered. Since then, I decided to continue on the work which I previously wrote in the report of group 14. This was the entirety of Section 2 on Unikernels/MicroVMs. I have asked the members of group 14 to not use my work as I wrote it completely on my own. This too I have communicated with the TAs and faculty of this course.

Furthermore, I deleted all work performed by the other members of group 14 and wrote the rest of the report completely on my own. All work in this report was strictly written by myself.

Web services and cloud systems

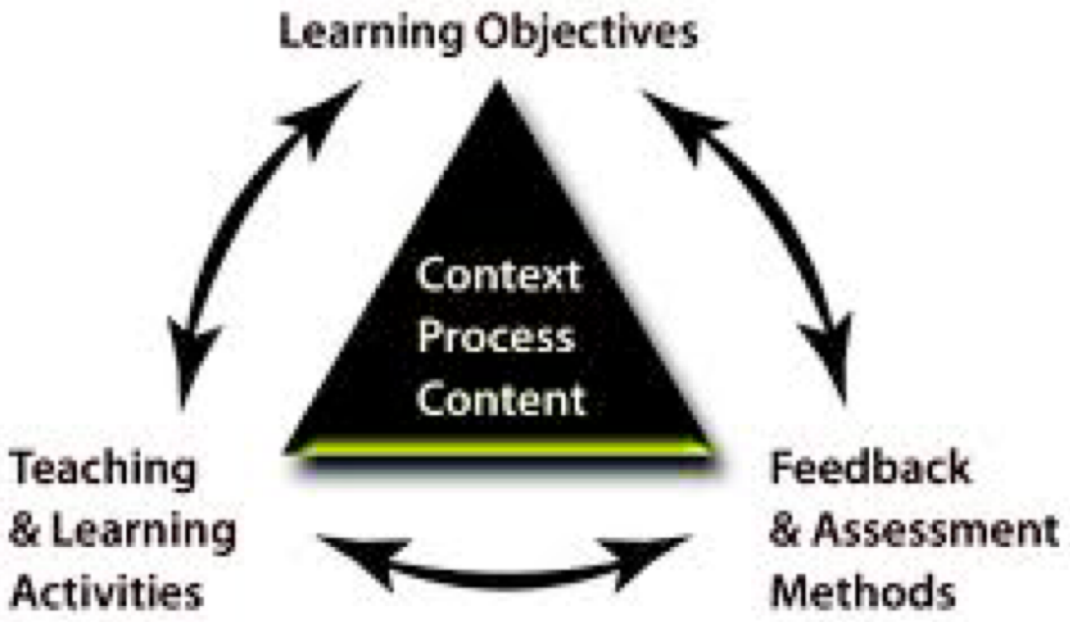
(MSc. Computer Science, VU-UvA)

Team: Adam Belloum + Guests

TA: Saba Amiri (contact for the Literature study and homework,

Onno Valkering, Reggie Cushing (contact for lab assignments),

Web services and cloud-based systems (MSc. Computer Science, VU-UvA)



Biggs, J. B., & Tang, C. (2011). *Teaching for quality learning at university: What the student does (3rd ed.)*. New York: McGraw-Hill Education (UK).

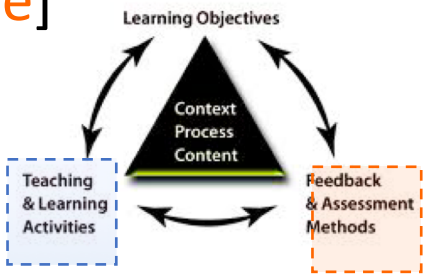
Bloom taxonomy [1]
Knowledge • **Lectures**
Comprehension • **Lectures**
Application • **Lab sessions**
Analysis • **Reading Assignment**
Synthesis
Evaluation • **Literature study**

[1] https://en.wikipedia.org/wiki/Bloom%27s_taxonomy

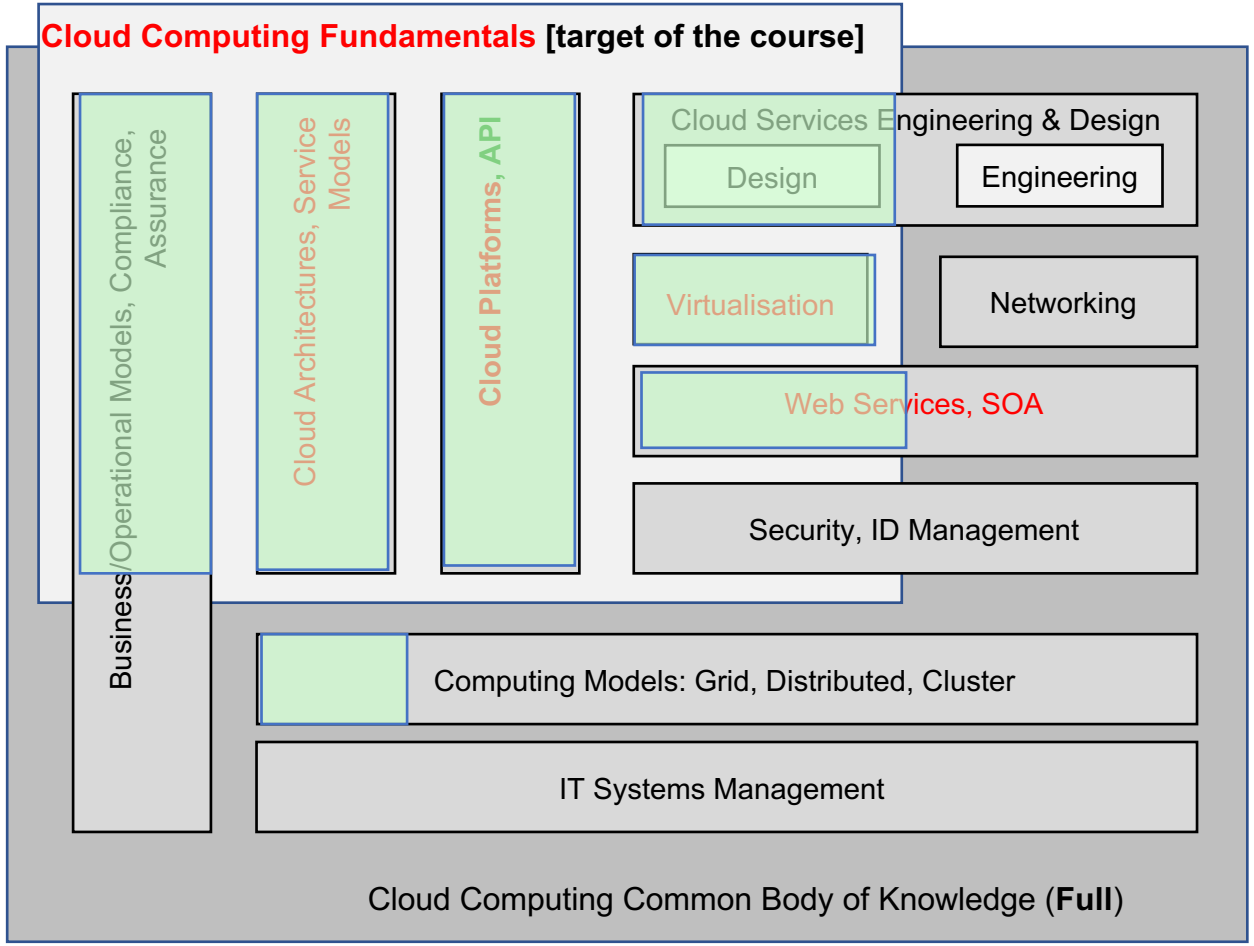
Web services and cloud-based systems (MSc. Computer Science, VU-UvA)

- Learning Objectives:
  - You will learn Cloud Computing as a new approach to distributed computing and background technologies and standards – homework Discussion and summary [feedback(1), 20% of the final grade]
  - You will develop practical skills which will help you to work with Cloud-base systems - practical Lab assignments [feedback(2), 45% of the final grade]
  - You will develop the ability to analyse scientific publications on cloud related topics - Literature study - [feedback(3), 35% of the final grade]

(1) Group feedback during lectures discussions or posted in canvas.
(2) personal feedback are given during Lab session (by TA),
(3) posted in canvas or email (lecturers and TA),



Common Body of Knowledge in Cloud Computing



Bloom taxonomy

Knowledge
- **Lectures**

Comprehension
- **Lectures**

Application
- **Lab sessions**

Analysis
- **Reading Club**

Synthesis

Evaluation
- **Literature study**

Y. Demchenko, D. Bernstein, A.S.Z. Belloum, A.M Oprescu, T.W. Wlodarczyk, and C. de Laat, New Instructional Models for Building Effective Curricula on Cloud Computing Technologies and Engineering, In Proceedings of the IEEE International Conference on Cloud Computing Technology and Science, Bristol, Dec 2013

Schedule

Weeks		Lecture topics	Labs
Week14	March 30 & Apr. 2 [Adam]	• Service Oriented Architecture • Web services SAOP and REST	SOAP & REST Services
Week15	April 6th & 9th [Adam] AWS SUMMIT RAI Amsterdam[2] 06/04 Deadline[1] Assignment 1	• Virtualization: Virtual Machine • Cloud platform	Microservice architecture New
Week16	April 16th [Adam]	• Modern distributed systems (Grid/Cloud/…)	Microservice architecture (Private Cloud →SNE resources)
Week17	April 20th & 23th [Adam/Reggie] 17/04 Deadline Assignment 2	• Containers and Container orchestration • Cloud architecture **IaaS**/Paas/SaaS • Cloud Standards	Container Orchestrations (Private Cloud →SNE resources)
Week18	April 30th [zhiming]	• Security in Cloud • DevOps/CD/CI	Work on your literature study
Week19	May 07th [Guest] 01/05 Deadline Assignment 3	Guest Lectures EU Cloud-based project and HPC cloud Work on your literature study prepare final report & presentation slides	
Week20	May 11th & 14th	**Student presentations**	
Week22	May 28th	**Student presentations**	
	Week 8 (TODO)	**Student presentations & Submit final report of the presentation slides**	

NOTE [1] All the deadline are fixed at Fridays midnight

[2] April 8th Amazon is organizing the Annual AWS Summit in RAI Amsterdam CANCELLED

International events organized in Amsterdam

- 2020 AWS Summits April 8th , 2020 CANCELLED

Online event

Activity 1: Homework

Quiz during the Lectures ???

1. kishore Channabasavaiah and Kerrie Holley, IBM Global Services, and Edward M. Tuggle, Jr., IBM **Software Migrating to a service-oriented architecture**
2. C Pautasso, O Zimmermann, F Leymann **“RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision”** Proceedings of the 17th international conference on World Wide Web, 805-814
3. Adriano Vogel et al. “**Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack”** 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2016
4. Foster I., Zhao Y., Raicu I., Lu S, **Cloud Computing and Grid Computing 360-Degree Compared**

Activity 2: Assignments

- Assignment 1: develop SOAP/REST service
- Assignment 2: Create a Microservice architecture
- Assignments 3: Virtualization using Docker & container Orchestration

Activity 2: Assignments

What you should know about the Lab infrastructure…

- We DO NOT OWN the infrastructure, neither have FULL CONTROL it → Things might(will) go wrong
- Software platform used in the Labs are very COMPLEX and still under CONTINEOUS development

  → We might not have a straightforward answer to some problems

- One group per topic
- Report + presentation
- Cross review + grading*

Activity 3: literature study

1. Servless computing (lambda function), FaaS (functions as a service)
2. IDaaS (identity as a service)
3. HPC and Cloud (potentials, challenges, and limits …)
4. Big Data and Cloud
5. Edge computing and relation to Cloud computing
6. fog computing and relation to Cloud computing
7. Distributed cloud-based ML/DL workflows
8. Machine Learning/ Deep Learning in edge/IoT devices,
9. Datacenter architecture models for clouds - Hyperconverged architecture,…
10. Mobile clouds: service models, infrastructure requirements, existing platforms and applications analysis
11. Multicloud management considerations
12. Cloud-based integration - iPaaS (integration platform as a service)
13. Payment systems in clouds (models used by Cloud Service Providers) and for cloud based applications
14. Impact of GDPR on Cloud computing in Europe- Data and user information privacy protection in clouds
15. Cloud-native, NewSQL, and globally-distributed databases - Google Spanner, Azure Cosmos DB, consistency trade-offs, local/zone/multi-zone replication, ...
16. MicroVMs and Unikernels for the Cloud. - AWS Firecracker, MirageOS, ...
17. Managing and monitoring Cloud resources
18. Infrastructure as Code, DevOps, Metrics: avg cost per customer, latency, ...
19. Cloud security practices - SecOps, ISO compliance, Audits, AWS GovCloud, ...
20. The European Open Cloud and similar initiative in Asia and US
21. Business model for Cloud computing

< you can also propose your own topic>

– need to be approved contact The TA (Saba) before you start to work on your any new topic>

What do you know about Cloud?

- Lets start with TV commercials
  - Microsoft “Winning Edge”
  - IBM, “Through the Cloud”
  - IBM, “Smart Cloud”
  - Western Digital “Personal Cloud”
  - HP Cloud systems “Featuring Charles Barkley”
  - SAP, “Run like Never Before”
  - Verizon, “Powerful Answers: Firefighters”
  - CenturyLink “Weekdays”