Literature study assignment

Coordinators: Saba Amiri, Adam Belloum,

1. Cloud Computing Business Models ……
2. Identity as a Service IDaaS ……...
3. HPC and Cloud (potential, Challenges and Limits…….....
4. The territorial Reach of the GDPS and Cloud Services…….....
5. The Impact of GDP on Cloud-base Healthcare systems…….....
6. Distributed Transaction in Microservices ….……
7. Comparative study of the state of the Art Tools for Cloud monitoring ……
8. Getting Grip on Technical Complex systems ……..……
9. The Impact of Edge and Fog Computing on IoT Applications for Smart Homes ……
10. State of the Art of Single Sign-on…….
11. Blockchain payment systems on Cloud ……
12. Distributed Neural graphs: DistDGL vs GraphTheta vs AliGrah …….
13. IDaaS - Research on Privacy u enhancement methods ……..……
14. Serverless computing and Function as a Service: security perpective .....……

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

UNIVERSITY OF AMSTERDAM

ASSIGNMENT 4A

Cloud Computing Business Models

June 3, 2022

Students:
Clifton Roozendal
10541039

Kasra Imanirad
13517066

Shashank Athreya
14130289

Group:
Assignment Group 3

Course:
Web Services and Cloud-Based Systems

Course code:
5284WSCB6Y

Abstract

The advent of Cloud computing removed the need for companies and consumers to invest in and maintain computer hardware. With a simple Internet connection, the users can directly store, access, and run programs on the Internet. Cloud computing can provide users with a reliable, customized, and cost-effective manner in various applications. However, as the cloud computing market matures and the competitors rise and fall, how do the market leaders define themselves enough that users prefer them over their competitors.

1 Introduction

The global public cloud computing market is expected to grow to 495 Billion U.S. dollars in 2022. This amount is made up of business processes, platforms, infrastructure, software, management, security, and advertising services made possible by cloud services[23] and in looking at figure 1, it might hit the 600 billion. This massive market is dominated by Microsoft Azure, Amazon Web Services (AWS), IBM Cloud, Google Cloud, and Alibaba Cloud.

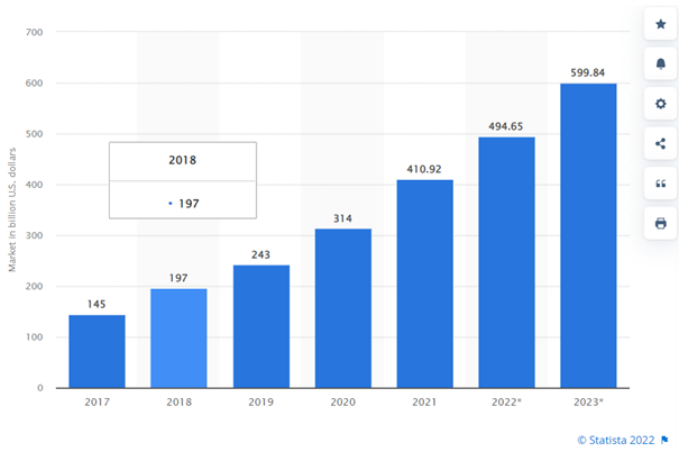


Figure 1: Worldwide forecast for the cloud market[24]

This paper will focus on the cloud business models and the interaction of actors in the cloud market as it keeps growing. As shown in Table 1, one must wonder how cloud providers and cloud users interact. For example, IT professionals at the organization that just switched to one of the cloud services will be challenged by cloud computing as it might differ from the skills they already possess [14]. Or for example, Amazon with its a controversial policy regarding its customers' data [25]. We will try to discuss these subjects in the paper. In chapter 2, we give a historical overlook of cloud computing. In chapter 3, we will discuss the service types and the business model of cloud computing. Chapter 4 will discuss consumer and provider interactions in the cloud market. In chapter 5, we will discuss the case study of Amazone as a surveillance breacher. At last, we will discuss our findings and conclusion.

2 History of Cloud Computing

It all started in 2006 when the term "Cloud Computing" was chosen while tech giants like Google and Amazon had a massive investment in implementing the applications on the web. They aimed to offer such service with fast deployment and constantly updating at the same time.

The evolution of major technologies and concepts such as cluster computing, distributed systems, and utility and grid computing has resulted in the birth of the cloud computing paradigm. Also, there is an ever-increasing interest in cloud-based systems because of the many benefits it has brought to the computing industry. Benefits include the rapid decrease in hardware cost, increase in computing power and storage capacity, overflow troubleshooting, growing size of data in the scientific area, and widespread adoption of web 2.0 applications.

Therefore, due to this evolution, the cloud concept became viable at the enterprise and consumer levels. Very soon, worldwide adoption and efficacy of cloud-based services rapidly increased over a short period. Consequently, major cloud providers reached a point to find a way to satisfy the rising demand. [7]

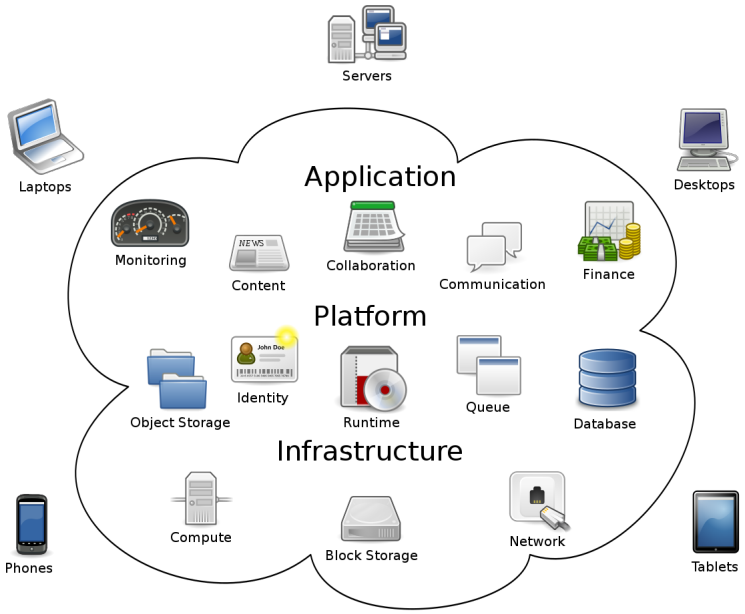


Figure 2: Cloud-based system[1]

Therefore, cloud computing can be considered as a result of evolution of various phases during the history of computing. For instance, Mainframe computing had been created in 1951 is

very reliable and powerful type of computing. It is responsible for handling large data operations like bulk processing such as online financial transactions. So such machines are working constantly without any interruptions and also high fault tolerance. By developing distributed computation and advent of mainframe computing machines, we witnessed a noticeable increase in system's processing capabilities.

Nevertheless, because of high expenses in implementing mainframe technology, cluster computing came up as a solution. In the sense that all the nodes in the cluster were linked together via a high bandwidth. By doing so, a cheaper network has been created which it was also capable of great computations. Thus, the expenses had been decreased to some certain extent but still the problem of geographical restrictions maintained. To address this issue, grid computing came up as a solution.

In 1990, along with growth of the internet, the paradigm of grid computing was introduced. By support of the internet, different systems were located at various location across the world and all connected to each other through the network. Since these systems were owned by organisations, the grid comprised of heterogeneous nodes and as an increase in distances, new problems had been emerged such as lacking of sufficient high bandwidth connectivity. Therefore, cloud computing marked a turning point in the history of computing and named as a successor of grid computing.[9]

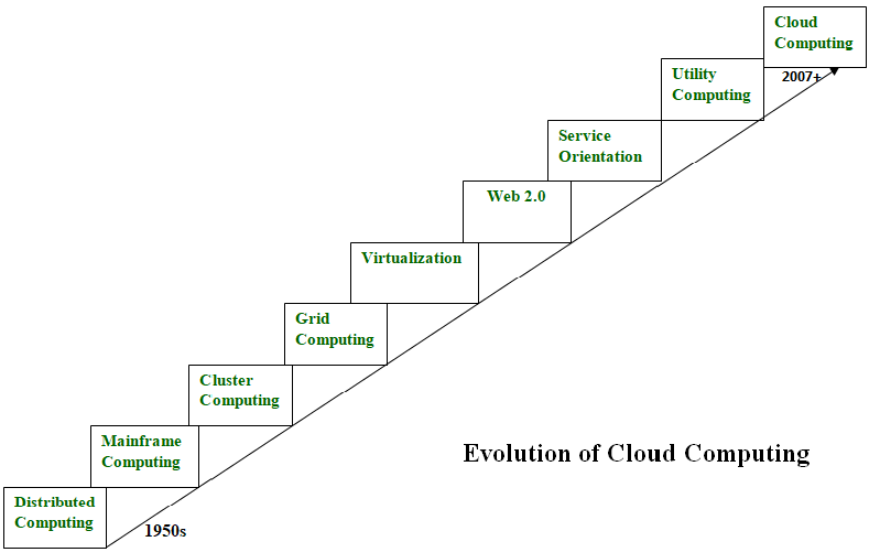


Figure 3: Evolution of cloud computing[2]

3 Cloud Computing Services

By growth of cloud computing and offering it as services to consumers, this paradigm has become as a service-oriented architecture that afterwards it became well-known as "Everything as a Service (Eaas)[8]. Based on this service-oriented architecture, cloud providers classified their services in three standard models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). This approach is also called the cloud computing stack because they are built on the top of one another.

There is also a pipeline which elaborates which components are running by the provider and which is offered to the customer. So the cloud provider in each model is taking care of different parts of the pipeline which determine some of its features such as flexibility, customization ability, security, handling the data, etc. Therefore, considering these features,customers at any level by choosing the proper type of service can leverage their cloud-based infrastructure or utilize the application and any other services. We will discuss it more in the following sections:

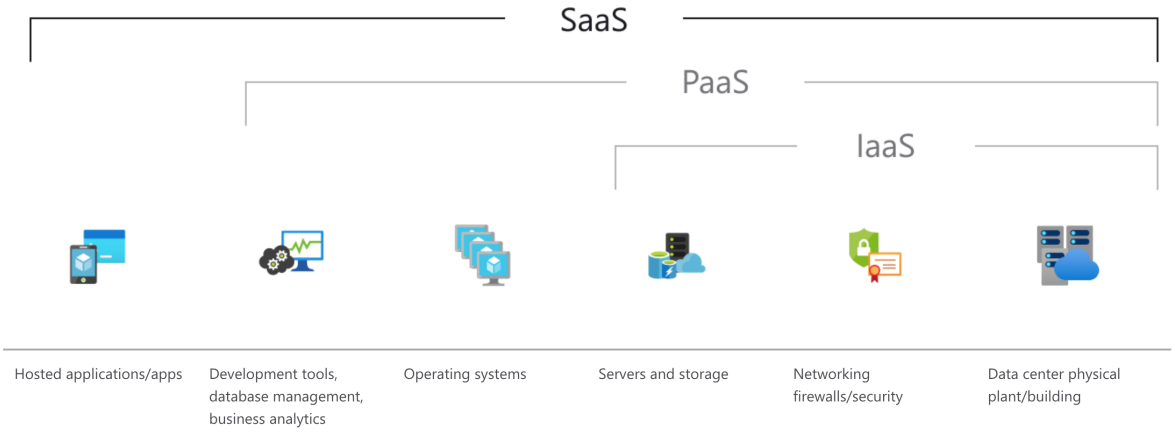


Figure 4: Evolution of cloud computing[2]

3.1 SaaS

When it comes to SaaS, it has become a huge industry because the paradigm of software over the last two decades has been changed completely. So we moved from an on-premises software development framework to instead a much more agile software development framework where they do not need to host such service on their own premises. So any application that companies or startups are offering today can be externalised through Software as a Service. Based on figure 3, cloud provider is taking care of the whole pipeline as back-end such as operating systems, servers and storage(data) and others except the application that is supposed to offer to the user. Therefore, it has minimum ability to customize which is known as flexibility feature, unless the user is at an enterprise level and in that case, the client can ask for way more customizations. Also, in SaaS model there are more limitations because the user has no access to the data.[19]

SaaS model deliver its services through web-based applications which are accessible via web browsers. Prominent examples would be Gmail and Google Docs which are accessible through various smart devices that can reach us to the conclusion that accessibility is one of the main advantages. Furthermore, the user is using does not need to buy the license, upgrade constantly, or run the application on their own premises.Therefore the major benefits of SaaS model would be: high scalability, accessibility, flexibility, etc.[12]

3.2 PaaS

As it can be seen in the figure above, in PaaS, there's a bigger part of the pipeline that is offered to the consumer. So cloud provider offers the platform to customize or create applications upon the request of consumer/user at any level which offers more freedom to the user to build their

desire application with particular functionalities. Also, the user is having the ability to possess the related tools on-demand without the need to build them on their own systems and meanwhile they can manage such applications and related data.

As a matter of fact, PaaS has been offered a perfect deployment and development environment and gives the customers the ability to develop simple cloud-based applications and also cloud-enabled applications at enterprise level. So the customer will purches the resources based on their needs (pas-as-you-go basis) and these applications are accessible from anywhere. Based on the figure above, Paas comprises infrastructure, servers, storage, development tools, middleware, business intelligence services, database management systems, etc. So as a powerful model, it supports the whole lifecycle of a web application such as: building, testing, deploying, updating and management. It also prevent some complexities and expenses of licenses, container orchestrators like Kubernetes, middlewares and apps, and other resources. Therefore, the customer manage the applications that developed based on their own preferences and cloud service providers will be taking care of everything else.[3]

3.3 IaaS

Infrastructure as a Service could the most flexible model because the the biggest part of the pipeline shown in figure above is possessed by the customer. So it offers essential storage, computation and networking resources on demand and it is based on a pay-as-you-go revenue model.

Migration the infrastructure to IaaS will eliminate the maintenance of an on-premises data center, reduce the expenses of hardware, and a good insight into business analysis. So it provides networking, storage, virtualization and hosting and as a result the user can leverage the cloud-based infrastructure without building it on their own premises. Also, NIST's has a definition of IaaS worth mentioning: "where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)."[18]

3.4 Cloud business model

By offering cloud services, cloud computing created an entrepreneurial ecosystem, through which vast majority of companies (in both business-to-business and business-to-consumer) are leveraging the cloud-based services that could result into a revenue consisting of precious products and services. So Being familiar with their concepts and their differences, could lead us into accomplish the business goals.

There is a wide range of commercial purposes by business models. For instance, B2B-enterprise companies that are doing large-scale data analysis, inventory management, business intelligence, etc. and B2C companies that are providing social media platforms, streaming services to customers. There are several revenue for such business models that have been built on the top of the cloud and primarily is subscription driven and and they are mostly on-demand services:[7]

- Subscription-based
- Consumption-based (pay-as-you-go)
- Advertising-based
- Hybrid models

4 The Cloud Computing Users and Competitors

4.1 Cloud Computing Consumers.

Cloud computing is a topic of interest for many information systems managers. More businesses have recognized the benefits of cloud computing and service orientation in cost and scalability. Not all corporations have the skills, organizational structure, and processes to realize this promise. IT professionals' capabilities will be challenged by cloud computing as the strong market growth of cloud computing and the fact that technology continually advances quicker than organizations can adapt to it[14]. Adoption of difficulties of cloud computing is present for users such as the following in table 1.

Cloud Challenges	Explanation
Availability/Reliability	Users have the expectations once they move everything to the cloud, are access to its services, the overall performance, and safety measures in case of failures.
Security and privacy	Security and privacy are a concern for users because third-party services and infrastructures are used to host data and perform operations.
Vendor Lock-in / Portability / Interoperability	A primary concern of cloud computing users is about having their data locked in by a particular provider.
Compliance/Regulatory ambiguity	Enterprise users must maintain business legal documents and ensure their integrity to comply with various national and internal laws. Cloud computing providers must adopt technologies and security measures to ensure that their enterprise users' data satisfy their compliance requirements.
Integration / Componentization	Integration with the existing architectures. The availability of tools enables the integration and componentization of applications.
Limited scope for customization	Users want cloud services to better adapt to their specific business model.
Vendor Management	Cloud computing has unique vendor management challenges and criteria to evaluate when considering strategic sourcing models and analyses.
Cultural Resistance	Cloud users might face organizational problems as it changes the tasks of their IT department, and the organization might not be prepared for the transition to the cloud.
Transition and Execution	Specific applications may not be suitable for use in the Cloud environments.

Table 1: Cloud adoption challenges[17].

Managing these systems requires a management approach that considers both the virtual and the physical sides. This is hard for IT managers who are used to conventional data centers[10]. To manage these difficulties, as stated in table 1 and migrate to the cloud with success, IT Teams must have their skills in cloud architecture, development, implementation, and operations. To solve these difficulties through competency development programs that encompass all

the organization's processes and the employees to sustain and improve the employee's competencies[17].

Challenges	Competency	Emerging Role
Availability/Reliability	Specific details of the service provided by the cloud provider should be done, including the availability and the minimum performance guarantee. The management processes would need to change to the new reality of moving into the cloud. Furthermore, planning for cloud outages needs to be prepared.	Provisioning Manager
Security	Additional challenges to making cloud computing environments as secure as in-house IT systems.	Security and Compliance Manager
Portability/Interoperability Integration/ Componentization/Customization	Manage the cloud platform for end-to-end business services by bridging the technological domains.	Cloud Architect
Vendor Management/Lock-in	Manage the relationship with the cloud service providers and integrate them into the existing services management and delivery processes. The IT department's role is changing from being internal on its own to partly or mainly managing external service providers.	Vendor Manager
Cultural Resistance	Change of management and managing the development of practical, educated, and career skills.	Training Manager
Transition and Execution	Assess essential business goals for cloud migration. Manage the cloud environments for specific business models. This is achieved by managing the cloud environments' configuration, operation, and performance.	Cloud Analyst

Table 2: Proposed competencies and roles[17].

However, these solutions are primarily managerial, as seen in table 2. The organizations adopting cloud computing should try to implement them to have a smooth transition towards the cloud.

Cloud computing is easier and cheaper for small and medium companies. However, it is not always the case, and most startups mainly achieve those cost savings instead. This is because those already established organizations likely already invested in the IT infrastructure. Therefore, it is financially more worth it to transfer to the cloud after the IT infrastructure is at the end of its lifecycle or is too obsolete[14].

Another thing that is underestimated is transaction costs. Transaction costs are associated with monitoring, controlling, and managing transactions. A transaction occurs when a product or a service is transferred between the vendor and the customer. When the customer buys a service or product from the vendor, the customer pays a certain amount to the vendor. There are always at least two parties involved in a business transaction. However, the amount used for buying the product or service should not be only considered when looking at the total cost[16].

The aspects of cloud computing transactions, namely cloud assets specificity, cloud uncertainty, and cloud transaction frequency, are shown in fig 5 and will be explained in the following paragraph.

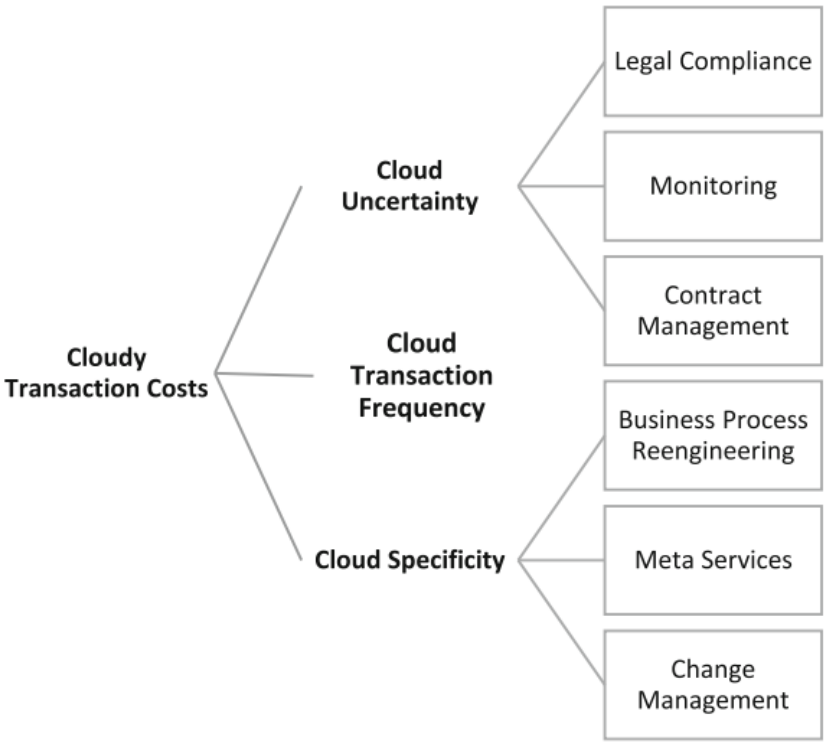


Figure 5: Cost areas based on cloud transaction dimensions[15]

4.1.1 Cloud specificity

Cloud specificity is based on (1) business process reengineering, (2) meta services costs, and (3) Change management costs with technologies[15].

Business process reengineering (BRP) is considerable and underestimated despite being complicated and vital for a cloud project's success. Business process reengineering is needed when an organization plans to do cloud adoption, but undeveloped processes might disqualify them from transferring to the cloud. Moving to the cloud would incur tremendous costs that may put the business at risk in such circumstances[27]. Redevelop the organization management according to the organization process rather than each functional department breaking the traditional model and emphasizing the business process's overall innovation to achieve the best results[15]. While being considered a non-IT business process that requires reengineering, billing and cost allocation should also be considered. Cloud users face different prices and service packages from vendors[5]. How can the accounting department split the costs across other departments of an organization? As each department has different consumption storing and computing, dividing the cost equally among departments wouldn't be fair[15].

Services that exist because switching to cloud computing are termed meta services. Meta services costs occur when organizations have to buy third-party services such as training courses, extra security services, or build their solution to manage their cloud portfolio [15]. For example, https://cloud-costs.com/, which was before known as Azure-costs.com, is a third-party service that helps with Cost management and optimization that is not offered by cloud providers such as Microsoft. Therefore organizations have to pay separately for it[22].

Change management costs are the many expenses such as paid training, hiring skilled cloud IT'ers, creating new strategies, and firing former qualified employees with outdated skills[11]. Not only that, but the chief information officer (CIO) has a vital role in the transition to the cloud, as he will change the organization's practices and mindsets. The CIO needs to be flexible and agile in his management duties[15].

4.1.2 Cloud Uncertainty

Cloud computing has always had a certain amount of uncertainty involved. Cloud uncertainty is based on legal compliance, monitory, and Contract management [15]. The main barrier for potential users is trust in the cloud provider. There needs to be a trusting business relationship between consumers and cloud providers, and the users need to have a dept understanding of the cloud provider's risk and accountability[6].

Legal compliance of data was already before the massive growth of the cloud market controversial. Cloud computing and the potential that data might leave national borders with different regulations open much potential unforeseen harm[4].There is a lack of knowledge in the legal department to work on cloud-related issues example, data residency. This lack of expertise is in the law community in general as the legislation around the cloud is still evolving continuously[26].There is a need for external legal consultancy to investigate compliance with laws. This will be an ongoing cost since its legislations are constantly changing[15].

Monitoring should always be done in every organization, even if they aren't using the cloud. However, monitoring the cloud requires cloud-specific monitoring tools. The reason is that departments of user organizations may use more services than needed while being unaware of the costs. These costs can end up being very high. However, then comes the question of which department should monitor it? It comes down to three entities in the cloud setup, the cloud vendor, the consumer IT department, and the cloud end-user. It is best left with the IT department, and the IT department will monitor it through the service level agreement. It is best to have some form of internal benchmark to compare the usage with, and the IT department has to have some form of measures to put limits and thresholds on the end-users before the costs get too high[15].

Cloud computing is marketed by providers and seen by users as an uncomplicated option with multiple advantages and is easily adapted. However, in reality, moving to cloud computing is harder. Transaction costs are the time and effort for negotiating, reaching out, contracting, and sustaining business relationships. Contract management is unavoidable for cloud consumers. The cloud consumers are usually neither ready for contract. Especially because contracts are put to govern a technology that is evolving and is changing by nature and the cloud consumers don't have experience managing such a field of law. To solve this the IT department and the law department should work together in creating and improving this[17].

4.1.3 Cloud Transaction Frequency

Cloud transaction frequency means how often services are implemented and how often a service is used. How often the services are used for a specific implementation, the more it is paid off to have it. Cloud transactions have high frequencies, in particular IaaS and Paas. The pay-as-you-go supports this, and the cloud facilitates scalability. This shows that cloud migration is usually worth it, even with transaction costs for users. However, they need to be well prepared to execute it[15].

4.2 Comparing Competitors

The cloud market is dominated by three multinationals Google, Amazon, and Microsoft. Figure 6 shows that Amazone dominates the market with thirty-three percent, twenty-one percent, Microsoft second place, and Google in third place with ten percent.

4.2.1 Service

Amazon is the largest provider and the pioneer in the Distributed computing market. Amazon AWS is dominating with design and monitoring. For the most part, AWS has a preference with organizations for its broad and massive contributions, venture helpful administrations, and open

and adaptable features. AWS and Microsoft Azure similarly offer a variety of groundwork, process capacity, storing and organizing, etc. Azure excels in its registering power, making it possible to send and manage virtual machines on a scale at whichever time limit. Azure has this over Amazon and Google. Azure also can be easily incorporated with other Microsoft products and offers open-source backing. Google cloud has a unique feature that it can provide cloud administration for engineers. Google cloud gives open source support, discount offers, migration ability, and flexible contracts. Because Google's origins are from analytical groundwork, Google Cloud is very adaptable to scientific devices[13].

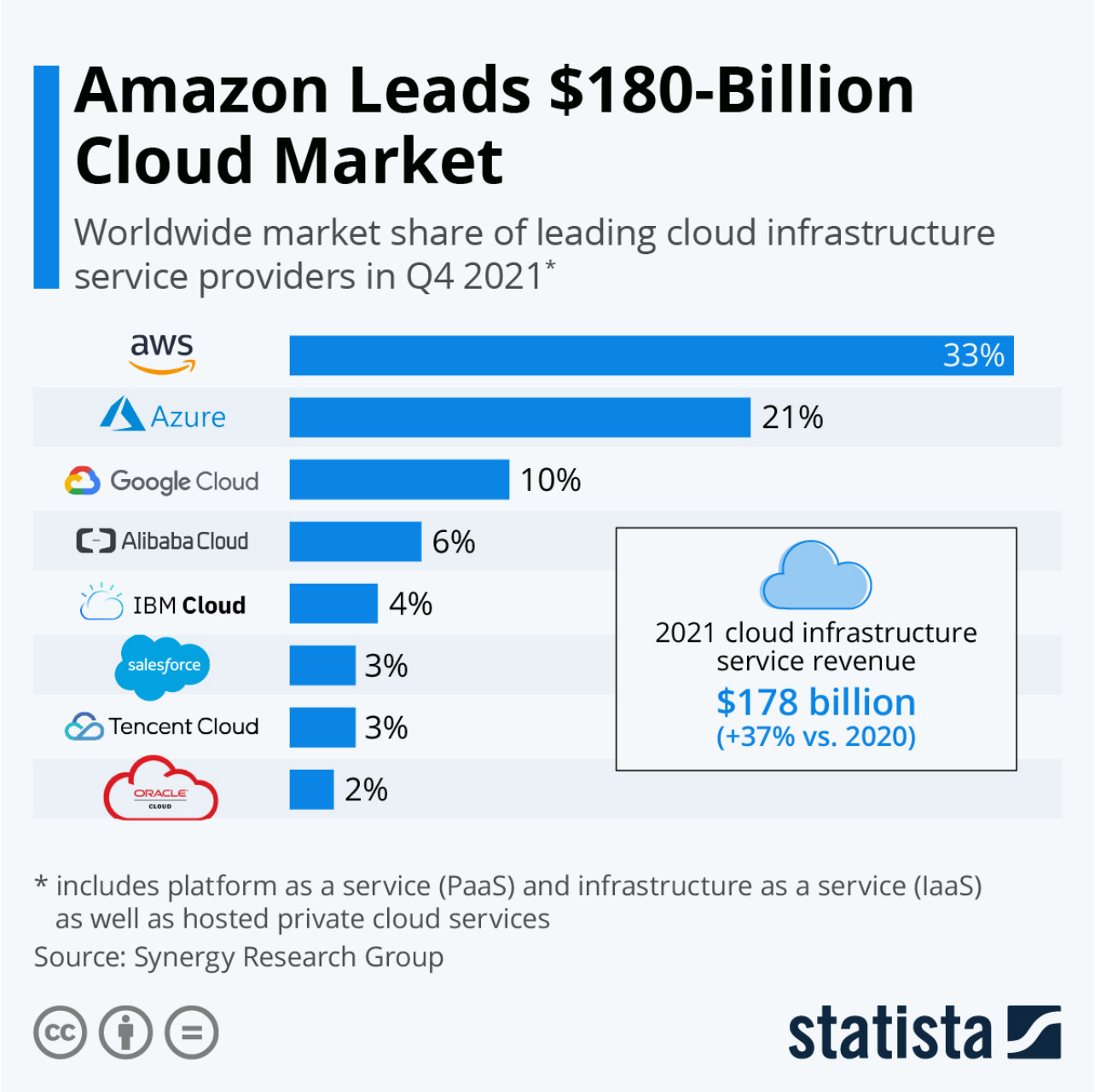


Figure 6: Market share of PaaS and IaaS[20]

4.2.2 Price

Amazon AWS offers every hour installment. Amazon gives free arrangement constrained capacity trial membership, which could attract potential people and organizations before they buy. Amazon payment plan based on monthly installments based on used hours. This can be very cost-effective for users, and likely gives AWS an Edge. Microsoft Azure offers two payment plans, pay ahead of time or a regularly scheduled installment plan. Google Cloud has the best cost-effective measure among the top three because it charges per minute or second of usage[13].

5 Case study: Amazon: Surveillance as a Service

It's important to consider the multiple perspectives on cloud and its providers. This section highlights the perspective of business and privacy advocates, on their view on amazon. To ease the analysis in this section, we define platform as a digital intermediary that allows interaction between multiple stakeholders; example, manufactures, distributors, retailers and consumers.

5.1 The business logic of Amazon's platform

Amazon is considered a platform provider, connecting buyers and seller, developers and end-users but in the view of digital economy, we notice their primary business is extracting and processing data [21]. To understand and process user activity and predict their behaviour allows in tackling the challenge of future UI/UX, development and services of the platform.

5.1.1 Alexa and her memory

The current focus on Amazon's Internet of Things (IoT) extends considerably to its well-known activity of collecting detailed data about its consumers. Amazon Web Services (AWS) promotes their Connected Home as a way for previously unconnected home devices such as appliances, lights, plugs, thermostats, doorbells, door locks, and home entertainment devices for the cloud. One can argue that Amazon sells surveillance as one of its services through these smart home devices.

The brand Amazon embraces the concept of collecting personal data from familial places as a crucial part of its customized relationship with individual consumers. It is a somewhat covert way of collecting data, which Amazon uses to offer surveillance a critical part of their way to provide personalized goods and services. Amazon orders its developers who seek to create capabilities for Alexa the following "Alexa should remember context and past interactions, as well as knowing a customer's location and meaningful details in order to maintain familiarity and be more efficient in future exchanges." Amazon has the will and ability to listen to and watch its consumers as part of its attribute[25].

5.2 Surveillance and Soft language

Advocates of privacy highlight a point of constant surveillance of user activity within these platforms has lead to a 'Surveillance Capitalism', coined by Shoshana Zuboff [29], where she further elaborates how the richest companies are not selling software but supervised platform to extract data. This coined term was later changed to platform capitalism and then to platform economy as it deals highly with the study of revenue generation though PaaS. Though it's renamed, it's functionality has remained unchanged, and this lead us to ponder why was it renamed?: in the previous section we highlighted how surveillance is offered as a feature and re-branded as 'learning', it's quite similar the case here. Platform economy sounds much less threatening than surveillance capitalism which induces correlation between data extraction and military surveillance [28].

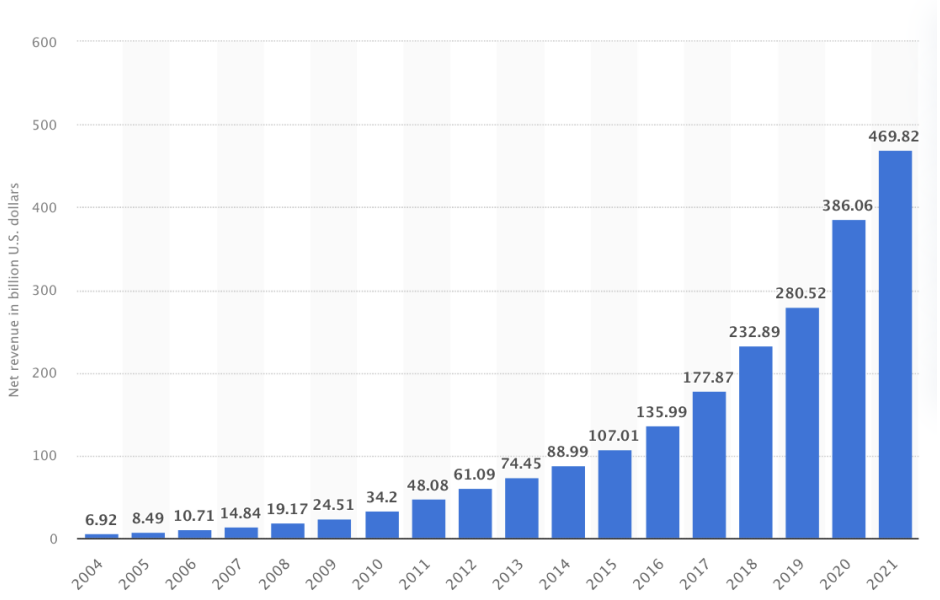


Figure 7: Amazon Revenue through the years

5.3 Business Interpretation

Although the case study highlights the legal invasion of privacy by Amazon, it also highlights its vast data interpretation. This is convenient for business executives to easily migrate to Amazon and their services in data processing. We also observe the reflection of the phrase "There is no such thing as bad publicity".Observing Amazon's revenue though the years in figure7, we notice significant jump in revenue during 2020 and 2021, we could speculate this to Amazon's data processing however, we lack sufficient research to conclude the exact reason for this jump.

6 Discussion

In this paper, we discussed many subjects such as the history, the business models, the consumer and provider behavior, and even a detailed case study on amazon's breach of privacy. There are a lot of concerns for users in the case of privacy, for organizations that want to grow, and even for the providers themselves. But even weighing this all, we believe that cloud computing is here to stay with the bad, the ugly parts, but also with its innovative and cost-effective sides.

7 Conclusion

The Cloud computing market is predicted to continue growing, as shown in figure 1 in the introduction chapter. How the market leaders continue to grow and compete will likely continue to change. We believe that the cloud market will continue growing despite the obstacles we mentioned for the providers and the consumers.

8 Division of work

Section	Responsible
Introduction	Clifton
History	Kasra
Cloud Computing Services	Kasra
The Cloud Computing Users and Competitors	Clifton
Case Study	Shashank
Discussion	Clifton,Shashank,Kasra
Conclusion	Clifton,Shashank,Kasra
Future Research	Clifton,Shashank,Kasra

Table 3: Division of work

References

[1] https://en.wikipedia.org/wiki/Cloud_computing#/media/File:Cloud_computing.svg. 2020.

[2] https://www.geeksforgeeks.org/evolution-of-cloud-computing/. 2020.

[3] https://azure.microsoft.com/en-us/overview/what-is-paas/. 2020.

[4] Susan Ariel Aaronson. "Data is different, and that's why the world needs a new approach to governing cross-border data flows". In: *Digital Policy, Regulation and Governance* (2019).

[5] Jonatha Anselmi, Danilo Ardagna, John C. S. Lui, Adam Wierman, Yunjian Xu, and Zichao Yang. "The Economics of the Cloud". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 2.4 (2017). ISSN: 2376-3639. DOI: 10.1145/3086574. URL: https://doi.org/10.1145/3086574.

[6] Erdal Cayirci and Anderson Santana De Oliveira. "Modelling trust and risk for cloud services". In: *Journal of Cloud Computing* 7.1 (2018), pp. 1–16.

[7] Gennaro Cuofano. "FourWeekMBA: Cloud Business Models". In: *Article* (2022).

[8] Yucong Duan, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud C. Narendra, and Bo Hu. "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends". In: *2015 IEEE 8th International Conference on Cloud Computing*. 2015, pp. 621–628. DOI: 10.1109/CLOUD.2015.88.

[9] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. "Cloud Computing and Grid Computing 360-Degree Compared". In: *2008 Grid Computing Environments Workshop*. 2008, pp. 1–10. DOI: 10.1109/GCE.2008.4738445.

[10] Roberto Gagliardi, Fausto Marcantoni, Alberto Polzonetti, Barbara Re, and Pietro Tapanelli. "Cloud computing for network business ecosystem". In: *2010 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE. 2010, pp. 862–868.

[11] Raoul Hentschel, Christian Leyh, and Anne Petznick. "Current cloud challenges in Germany: the perspective of cloud service providers". In: *Journal of Cloud Computing* 7.1 (2018), pp. 1–12.

[12] Andrew Joint and Edwin Baker. "Knowing the past to understand the present1 – issues in the contracting for cloud based services". In: *Computer Law Security Review* 27.4 (2011), pp. 407–415. ISSN: 0267-3649. DOI: https://doi.org/10.1016/j.clsr.2011.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0267364911000689.

[13] Muhammad Ayoub Kamal, Hafiz Wahab Raza, Muhammad Mansoor Alam, and M Mohd. "Highlight the features of AWS, GCP and Microsoft Azure that have an impact when choosing a cloud service provider". In: *Int. J. Recent Technol. Eng* 8.5 (2020), pp. 4124–4232.

[14] Angela Lin and Nan-Chou Chen. "Cloud computing as an innovation: Percepetion, attitude, and adoption". eng. In: *International journal of information management* 32.6 (2012), pp. 533–540. ISSN: 0268-4012.

[15] Rasha Makhlouf. "Cloudy transaction costs: a dive into cloud computing economics". In: *Journal of Cloud Computing* 9.1 (2020), pp. 1–11.

[16] Hasan Nuseibeh. "Adoption of cloud computing in organizations". In: (2011).

[17] John Otieno Oredo and James Njihia. "Challenges of cloud computing in business: Towards new organizational competencies". In: (2014).

[18] Timothy Grance Peter Mell. "The NIST Definition of Cloud Computing". In: *The NIST Definition of Cloud Computing*. 2011, pp. 1–10. DOI: 10.1109/GCE.2008.4738445.

[19] Aaqib Rashid and Amit Chaturvedi. "Cloud computing characteristics and services: a brief review". In: *International Journal of Computer Sciences and Engineering* 7.2 (2019), pp. 421–426.

[20] Felix Richter. *Infographic: Amazon Leads $180-Billion Cloud Market*. 2019. URL: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/.

[21] Nick Srnicek. *Platform capitalism*. John Wiley & Sons, 2017.

[22] app.limit UG. *cloud costs*. 2022. URL: https://cloud-costs.com/.

[23] Lionel Sujay Vailshery. *Public cloud computing market size 2023*. 2022. URL: https://www.statista.com/statistics/273818/global-revenue-generated-with-cloud-computing-since-2009/.

[24] Lionel Sujay Vailshery. *Public cloud services growth worldwide 2022*. URL: https://www.statista.com/statistics/203578/global-forecast-of-cloud-computing-services-growth/.

[25] Emily West. "Amazon: Surveillance as a service". In: *Surveillance & Society* 17.1/2 (2019), pp. 27–33.

[26] Ogan Yigitbasioglu. "Modelling the Intention to Adopt Cloud Computing Services: A Transaction Cost Theory Perspective". In: *Australasian Journal of Information Systems* 18 (Nov. 2014), pp. 193–210. DOI: 10.3127/ajis.v18i3.1052.

[27] Yuanxing Zhao and Junhong Gao. "Research on Accounting Process Reengineering Based on cloud computing in the era of big data". In: *2021 4th International Conference on Information Systems and Computer Aided Education*. 2021, pp. 521–525.

[28] Shoshana Zuboff. *The age of surveillance capitalism: The fight for a human future at the new frontier of power: Barack Obama's books of 2019*. Profile books, 2019.

[29] Shoshana Zuboff, Norma Möllers, David Murakami Wood, and David Lyon. "Surveillance Capitalism: An Interview with Shoshana Zuboff". In: *Surveillance & Society* 17.1/2 (2019), pp. 257–266.

Identity as a Service

Abdellah Lahnaoui
Department of Computer Science
Universiteit van Amsterdam
abdellah.lahnaoui@student.uva.nl
Student-ID: 13215566

Adrian Aabech
Department of Computer Science
Universiteit van Amsterdam
adrian.aabech@student.uva.nl
Student-ID: 14120429

Ruben Horn
Department of Computer Science
Universiteit van Amsterdam
ruben.horn@student.uva.nl
Student-ID: 13676091

Abstract

Identity is a central part of any multi-user system. With the move to microservices and the cloud, new challenges and opportunities arise for the task of identity management. Developments in this area could impact architectural decisions and further development of cloud based systems. In this paper, we motivate and investigate federated and decentralized identity as alternatives to isolated solutions. We perform an analysis of the literature on this topic using a publication search engine and select publications for two categories of approaches that build on federated protocols or decentralized technology. We motivate and establish a set of criteria to compare a selection of solutions and discuss our findings. We conclude that established federated protocols are still relevant and more innovation can be expected in the area of decentralized approaches.

1 Introduction

Digital identity management has a substantial history in Europe, where the European Union has funded different projects for the last 20 years [11, 12]. Commonly, they were concerned with national or regional log-on identity services [27]. The intention behind this is providing log-on services with reliability from the state, letting the nation or one of its institutions act as their trusted identity agent. This could allow citizens of a country to, for example, securely access sensitive information.

Cloud computing has become a wide-spread practice in the last two decades due to the provided speed, high availability, scalability, and security [7] which are benefits for businesses migrating to the cloud. With respect to identity management, several cloud vendors provide services for operating identity services in the cloud, such as AWS Cognito, Azure Active Directory or Google Cloud Identity [10, Chapter 9]. Users will have to adapt to creating and providing an online identity for each one of these cloud services [7] using the isolated identity providers, which correspond to a single service or distinct set of services. Therefore, in an effort to increase interoperability, the industry is moving towards using an IDentity as a Service (IDaaS) model [31]. This model permits selected external providers to manage and store users' personal data and allows them to authenticate and/or authorize with other cloud-based services [19].

These IDaaS models contain a variety of services, but typically include Single Sign-On (SSO), Multi-Factor Authentication (MFA) and Privileged Account Management (PAM) among others [9]. Today,

Web Services and Cloud-based Systems (UvA 2022), Amsterdam, Netherlands

the private sector is dominated by SSO services, like Google, LinkedIn, Twitter or Facebook accounts. Alternatively, service providers could require some alternate form of identification, commonly allowing users to create their own accounts.

However, as beneficial a such federated approaches are, they still have similar disadvantages to the isolated identity management. Users will most likely have to access resources from various domains and, in a real life setting, would still need to authenticate with multiple identity providers [41]. Accordingly, there is a considerable amount of ongoing research to come up with an alternative, such as a blockchain-based decentralized identity management [18]. In such a decentralized system, the user fully owns and controls their identity data and as such, they interact directly with the service providers, abolishing the need for a middle-man [27].

We are interested in finding out whether recent developments regarding identity management and emerging technologies can be seen as an indicator of the future of identity management, and which approaches are the most promising.

Consequently, this paper will explore the state of available literature to provide an insight into what is being used, researched and developed. Then, there will be an analysis of modern approaches to identity management, including notable solutions from academia and the industry. Afterwards, a comparison of these solutions is conducted to ascertain their relevance and their stance regarding various issues. Finally, the study will conclude with a brief summary and a look into future developments and the potential of identity as a service.

1.1 Terminology

Below, we give some definitions for key terminology used in this paper to describe key components and foundational concepts of identity management.

Identity An identity represents a single user and is used for the authentication and authorization. A user may have multiple identities for multiple or even the same service.

Attribute Identities do not have to be atomic and immutable. Additional attributes can be linked to an identity. They are characteristics, such as name, date of birth, profile picture of the user and so on, that defines the identity.

Identity provider An identity provider manages and provides user identities and additional user information and implements authentication mechanisms. In the context of cloud computing, such a standalone identity solution is referred to as IDaaS.

Service provider A service provider may rely on a user identity and ancillary information, which is obtained from an identity provider, for the purposes of, primarily, maintaining privacy and accountability through authorization as well as for billing.

1.2 Identity management technologies

In this section, we give a brief overview over the major authentication and authorization protocol standards and technologies. These standards are also widely used [46] in the implementation of identity federation, however, for the purpose of investigating future trends, we only consider solutions which extend them for comparison.

1.2.1 Kerberos

Kerberos [35] is a ticket based authentication system developed by the Massachusetts Institute of Technology for project Athena. It uses a Key Distribution Center (KDC) that is composed of two servers, the Authentication Server (AS) and the Ticket-granting Server (TGS). One implementation of the KDC is the Windows Domain Controller [22, Chapter 4]. Users authenticate themselves at the AS and receive a Ticket Granting Ticket (TGT) which can be used to obtain a ticket from the TGS to be used to access the protected resource on the application server. Tickets in Kerberos have expiration dates. Multiple organizations can be connected to share resources between their users by sharing the key of the TGS [22, Chapter 8.1].

1.2.2 SAML

Security Assertion Markup Language (SAML) is an eXtensible Markup Language (XML) based standard which allows identity providers to pass authorization credentials to service providers [2]. So, the user, using a browser agent, accesses a service provider which in turn redirects the SAML request back to the browser. The browser then relays the SAML request to the identity provider, which has the choice of using any suitable authentication method. The identity provider then generates a SAML assertion and sends it to the service provider through the browser agent. Finally, the service provider sends the security context to the browser and allows the user to access requested resources.

1.2.3 OAuth

Open Authorization (OAuth) is a standard that grants secure delegated access to a variety of services via access tokens [25]. This replaces the need for using credentials and enables users to securely grant access to their data to other service providers on their behalf. It also has the advantage of using JavaScript Object Notation (JSON) packets and API calls, which makes it more optimized with recent web and mobile applications (compared to SAML). A common workflow when using OAuth is as follows: The client requests an access token from the identity provider, which then redirects the user to its respective authentication and consent procedure. Afterwards, the identity provider returns an access token. The client can use this token with every request in order to access the resources needed.

1.2.4 OpenID

OpenID connect allows service providers to delegate the authentication of users to identity providers [3]. It uses OAuth as a foundation and extends it with an extra ID token which contains the identity claims in JSON format. OpenID's workflow is very similar to OAuth. The main difference is that OpenID takes care of the authentication, while OAuth handles the authorization. Consequently, the identity provider encodes the user details into a JSON Web Token (JWT) identity token that contains user information and signature. This token is then passed to the client, which confirms the JWT and confirms the signature using a public key.

1.2.5 SSO

SSO is an authentication method that allows users to use one set of credentials to authenticate to multiple services [14], possibly in a different organizational domain. A central service performs authentication and then shares the session with other services by, for example, sharing a signed JWT.

2 State of Publications

To further develop and understanding of the topic, we conduct a survey of the available literature. The goal is to develop an understanding of what has and is being researched, and how that reflects in the technologies being developed and used.
First, we compile a list of search keywords, which we identify after shallow manual exploration of the topic *Identity as a Service* using Google Scholar. On this basis, we conduct an automated search. We do this by utilizing the Semantic Scholar API [1] to search using the keywords referred to in Table 1 and using the category *computer science*. The keywords *identity and access management* and *identity management* are more general and are included for context. By doing this, we get the amount of publications and their keywords and can start developing an understanding of all the available literature in this corpus. All keywords are provided in lower-case, since the search is not case-sensitive.
We look for papers published after 1990 to potentially include early publications covering grid computing that predate the cloud, since identity management is relevant to other multiuser systems as well.

2.1 Limitations

The number of publications for the keyword *idaas* is not representative, since we are only able to obtain a very small subset of using the API of the roughly 11,700 publications as of June 3, 2022 that are returned by the browser search.

The official API for Semantic Scholar only returns up to 10,000 publications ranked in descending order of relevance [21] for any given query regardless of the actual number of publications matching this keyword and has a strict rate limit of 100 requests within a five-minute window. As of June 3, 2022, Semantic Scholar indexes roughly 200 million publications [1]. Additional sources such as Scopus and Google Scholar could be included to search among a larger number of publications. Scopus offers an authenticated API, but Google Scholar can only be crawled using third party solutions.
While results from the API query are filtered by field of study, this is too coarse and the results still contain false positives.

2.2 Literature in numbers

Among the 14827 publications that were retrieved 3292 (22.20%) were found for the keyword *decentralized identity management*, 3633 (24.50%) for *federated identity management*, 4638 (31.28%) for *identity and access management*, 2528 (17.05%) for *identity management* and 700 (4.72%) for *self-sovereign identity*. Only 36 (0.24%) were found for *idaas*, which is explained above.

Table 1 shows the average number of yearly publications and average number of citations per publication for each keyword. The keywords *federated identity management* and *decentralized identity management* are very similar with respect to publication per year and number of citations per publication. The keyword *self-sovereign identity* stands out as having particularly few publications, and the high mean and standard deviation of the number of citations can be attributed to one particular false positive.[1] The average and standard deviation of the number of citations is a bit more varied. The keyword *self-sovereign identity* has both an especially high average number of citations per publication, but also a very high standard deviation.

Keyword	$\mu \frac{\text{publications}}{\text{years}}$	$\sigma \frac{\text{publications}}{\text{years}}$	μ **citations**	σ **citations**
federated identity management	95.61	89.24	22.47	136.13
decentralized identity management	78.38	82.08	43.0	178.32
self-sovereign identity	17.95	15.73	288.41	1391.12
idaas	3.0	1.76	14.61	40.5
identity and access management	118.92	121.34	15.36	51.02
identity management	64.82	67.8	34.73	91.98

Table 1: Keyword statistics

Figure 1 shows the number of publications per keyword over time in absolute numbers and relative to the total number of publications for the corresponding keyword, respectively. The drop in number of publications across all keywords can likely be partially attributed to the coronavirus pandemic starting in late 2019 [34]. Additionally, the search was performed in June 2022, so not all publications from this yer are accounted for and indexing may be somewhat delayed. The recent peak for the keyword *decentralized identity management* coincides with the maturing of blockchain and distributed ledger technologies, and is also a use-case for many such projects. The related topic of *self-sovereign identity*, however, does not seem to experience the same development. At the same time, publications on *federated identity management* seem to be plateauing. Generally, all topics grow in popularity up until the early 2010s and remain relatively high. We did not find an explanation for the recent spike in publications related to *identity and access management*.

[1] The result `https://www.semanticscholar.org/paper/a67568e046219eb705eaca2ee6630a76fc4b3043` (Accessed June 3, 2022) for instance falls under the category of *philosophy* and is not relevant to the topic of this survey, but was returned due to the high number of citations.

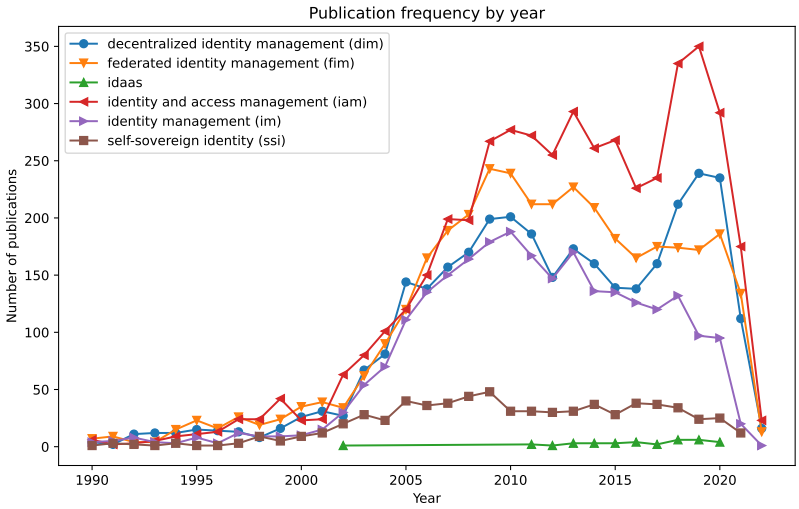


Figure 1: Publications over time per keyword

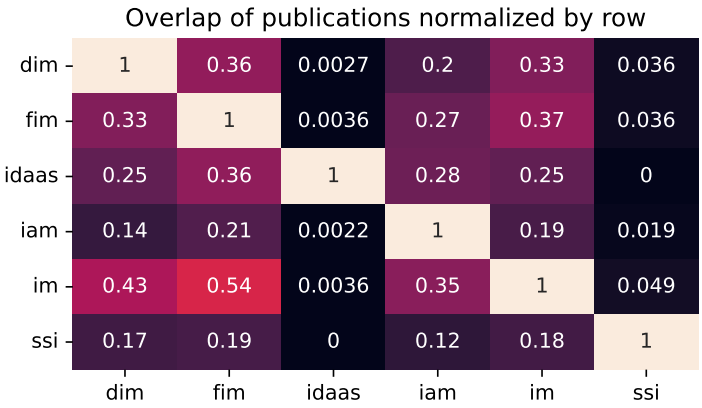


Figure 2: Fraction of publications covering two keywords

Figure 2 shows the fraction of papers for a given keyword in a row that are also related to a second keyword in the corresponding column. The keywords *decentralized identity management* and *federated identity management* have a reasonably high correlation, with 33% and 36% respectively. Among publications on *identity management*, these two keywords stand out as having a very high occurrence of 43% and 54%, but a significantly lower rate for publications on *idaas*. The topic *federated identity management* seems to be generally more popular, whereas *decentralized identity management* might 'catch up' given recent trends as mentioned before.

2.3 Relevant literature

While a query on a single keyword contains a considerable amount of false positives, the low number of results for *idaas* allows us to select relevant publications by filtering for those that also match this keyword, along with *federated identity management* or *decentralized identity management*. The resulting publications are listed in Table 2.

Paper	Search keywords	Content
Nuñez et al. 2015 [39], Nuñez and Agudo 2014 [36], Nuñez et al. 2013 [38], Nuñez et al. 2012 [37]	decentralized identity management, federated identity management	This series of papers proposes BlindIdM, a user centric IDaaS solution in which the identity provider can supply the user information without knowing it himself. This is facilitated through the use of a proxy re-encryption scheme which allows to re-encrypt the information for a different key without revealing it. This solution can be integrated with OpenID and SAML. This approach should eliminate the trust requirement between service providers from service providers to identity providers. The paper compares this solution to 'traditional' identity federation between organizations.
Fayolle et al. 2011 [20]	decentralized identity management, federated identity management	This paper makes a case for an identity centric approach for the future internet. They identify fragmentation as a key problem of the current state of identity on the internet. The proposed 'Identity in the Cloud'-Agent (IC-Agent) is proposed to solve this by facilitating the separation of data storage and exploitation. These agents are used in-line and in a symmetric fashion. IC-Agents can be self-hosted, they do not replace attribute validators.
Vo et al. 2019 [49], Vo et al. 2018 [48], Vo et al. 2017 [47]	decentralized identity management, federated identity management	These papers propose an architecture for a privacy supporting federated identity architecture based on purpose-based attribute-based encryption with SAML and OAuth. Attribute decryption is controled by two tokens which verify the time and purpose respectively.
Li et al. 2018 [30]	decentralized identity management, federated identity management	This paper proposes Sh-IDaaS, a discovery service model based on the Shibboleth [13] SAML implementation to facilitate multiple identity protocol through the use of a conversion layer. Identity providers can be dynamically added or removed.
Lee 2018 [29]	federated identity management	Blockchain Based IDaaS (BIDaaS) leverages a private blockchain which service providers can use to look up identities of users. Users are authenticated by a unique cryptographic challenge using which can only be solved using their private key.

Table 2: Relevant literature for *idaas* from API query

2.4 Additional literature

We use Google Scholar and Scopus to manually research other solutions in addition to solutions from the literature obtained by Semantic Scholar API, we also compare ShoCard, uPort and Sovrin, which are alternative Self-Sovereign Identity (SSI) solutions, in which the user fully owns and controls their identity, that utilize blockchain technology [18, 45].

3 Comparison of modern approaches to Identity Management

3.1 Isolated identity services

Isolated identity services follow the most basic approach to identity management, where each domain uses a separate identity service, thus requiring separate user credentials for multiple services from different domains. This is inconvenient for the end user as each user's attributes are managed in isolation, which leads to a massive number of identities where a lot of them are replicas [28]. Managing multiple login credentials can also overwhelm the user and lead to bad security practices such as password re-use.

3.2 Motivation for alternatives to isolated identity services

While isolated identity management services are a straight forward solution to identifying users of service providers, they also have several potentially undesirable implications. We found the following desired properties that a federated or decentralized approach could improve upon over an isolated identity service:

Portability Isolated identity services do not share or exchange user repositories. This means that a user has to interact with and manage credentials for multiple identity providers (e.g., Facebook and Google) in order to use different services [26]. It is not possible to transfer identities belonging to the same user, leading to fracturing and duplication of information across multiple services. This may also be seen as a privacy feature, although identity linking is a possible risk if the same identifiable personal information is used across multiple identity providers [8]. Users want to be able to interact with service providers without being forced to migrate to a new identity provider [15].

Unlinkability On the other hand, one prominent service provider may arise due to the increased convenience of only having a single identity, at which point service providers may be able to learn sensitive information about the user by tracking them. For Facebook's identity provider Facebook Connect [18] this may be the case. The corresponding property of unlinkability describes to which extent correlations that could reveal such data are hidden. This concerns both linking actions to the same identity (by a service provider) and linking two identities belonging to the same user (by the identity or by service providers colluding with each other) [8].

De-coupling Through tight coupling between service providers and identity providers, the number of users is limited to the users of the chosen identity provider. Each service provider involved must obtain user data from the identity service directly or from a service consumer who propagates it, which may be a privacy concern, because it is not controlled by the user [49]. Integrating multiple service providers from the side of the service providers results in considerable development and maintenance effort with no apparent benefit to the provided service, as well as all the disadvantages mentioned above. Termination of the identity provider would also render the service provider unusable, making them a single point of failure in the worst case.

Privacy If users want to interact with the service providers, they must also interact with and potentially disclose personal information to the identity provider, which may have negative implications for privacy. A central identity provider may have knowledge of all attributes of the user, and the access of a service provider to these attributes may not be minimized. Returning to the example of Facebook

Connect, an authorized service provider may continue to access user information within the same scope at will [8]. This issue is different from Section 3.2, since it does not require the combination of two or more data points.

3.3 Federated solutions

A federated solution, briefly explained, is the result of establishing connections between multiple identity management systems [16]. How it operates in practice is through users trying to access service providers, and the service providers contacts an identity provider to verify the users' identities. Therefore, on a technical level they are often constructed similarly to an isolated solutions, but the authentication is handled by a third party, using standards like OpenID or SAML for identification and OAuth for data access.

3.3.1 BlindIdm

Attributes in BlindIdm [39, 36, 38, 37] are end to end encrypted in this approach as visualized in the OpenID flow visualized in Figure 3. This approach is designed to eliminate the need for trust in the identity provider to allow outsourcing identity management which could be especially relevant for business moving to the cloud.

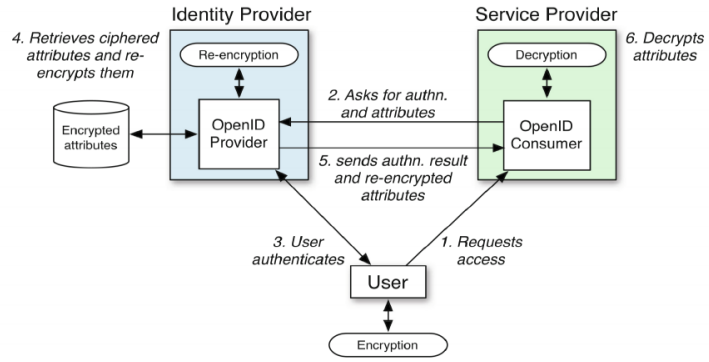


Figure 3: OpenID flow from Nuñez et al. [39]

Portability Portability is not addressed by this approach beyond SAML and OpenID. However, since the identity provider is not a trusted party, the validity of attributes is not tied to it and a user could choose to re-establish their identity with a different identity provider.

Unlinkability While the identity provider cannot decrypt user attributes it may be possible to infer them through the interaction with service providers in some cases with prior domain knowledge about the services and interaction models. For example the identity provider could determine the department of a user in a business from the type of service that they interact with like an ordering service implying logistics or a journaling service implying book keeping.

Loose coupling to service provider Through the use of SAML and OpenID, tight coupling to identity providers can be avoided. However, the concrete identity provider for the domain of the user must be integrated by the service provider.

Privacy Through the use of proxy re-encryption, the identity provider cannot learn any attributes about the user. Since in the scenario envisioned by this paper there is a direct trust relationship between the domain of the user and that of the service provider, but not between the service provider and identity provider, a collusion between identity provider and service provider can be seen as unlikely.

3.3.2 IC-Agents

This ambitious proposal by Fayolle et al. [20] suggests a separation of data storage from services which access and process it. Attributes can be verified through assertions within a web of trust. Access to data and services goes through user operated IC-Agents.
Beyond HTTP and onion routing, no concrete technology is proposed. The reliance on cryptographic operations and per-user services which are running constantly implies potentially significant overhead.

Portability Because IC-Agents are intended to be operated and can also be hosted by users, identities are portable by default.

Unlinkability While a users's IC-Agent may provide distinct identities for different services or actions, tracing the network connection back to the server hosting the IC-Agent could be used to link them. To mittigate this, it is proposed to use onion routing for all egress traffic. Alternatively or in addition to that, a user may run multiple IC-Agents which would prevent linking if all ingress traffic uses onion routing.

Loose coupling to service provider Since the user's interactions with a service are proxied through the IC-Agent, the identity of the user is provided by themselve from the view of the service provider. In order to verify user attributes the service provider must trust the certifier of a given certificate.

Privacy Because the IC-Agents, which store attributes, are operated by the users, and only requested information is disclosed to service providers, a high level of privacy can be achieved. If users must be de-anonymizeable, a trusted identity provider would issue a pseudonym. Although this is often not required, it is likely that all service providers will demand such assurances for legal reasons, which could severely undermine the users' privacy.

3.3.3 Vo et al.

Figure 4 shows the overview of the proposed solution by Vo et al. [49, 48, 47] on an example of three service domains. In this example, the user only directly interacts with one service provider. The encrypted personal information is deseminated through the users identity provider. Subsets of attributes can be decrypted using a unique combination of time based access token and policy derived token.

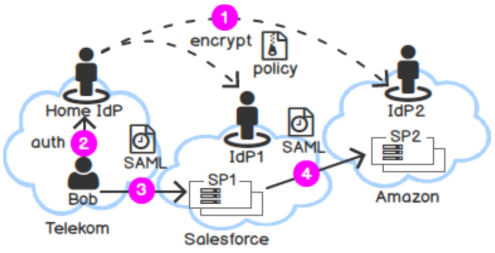


Figure 4: Overview from Vo et al. [48]

Portability Portability is not addressed by this approach beyond SAML and OAuth.

Unlinkability Trivial identity linking can be avoided, even if service providers colude with one another, if different attributes are disclosed.

Loose coupling to service provider Through the use of SAML and OAuth, tight coupling to identity providers can be avoided. However, the concrete identity providers must be integrated by the service provider. In the presented scenario, the trust relation between the domain of the service provider and that of the user extends extends to the identity provider, since it is part of the user domain.

Privacy This approach is very beneficial to the user privacy outside of the home domain, since access to attributes is minimized on an attribute and purpose level.

3.3.4 Sh-IDaaS

As visualized in Figure 5, the Sh-IDaaS [30] solution consists of a discovery service which acts as a broker to the user's identity provider. By means of a SAML converter middleware for the required identity protocol, any service provider can be integrated. This solution could be extended to facilitate inter-cloud identity management as suggested by [17], even if different protocols are used.

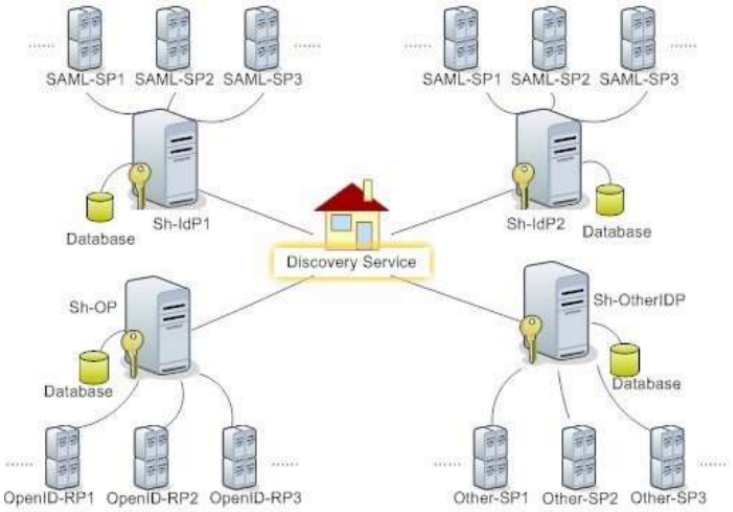


Figure 5: Architectural overview from Li et al. [30]

Portability Portability is not addressed by this approach, because this depends on the connected identity provider and the underlying protocol used before the SAML conversion.

Unlinkability Identity linking is not addressed by this approach beyond SAML.

Loose coupling to service provider Through the use of a discovery service, identity providers can be added or removed without affecting the service providers. However, the discovery service naturally provides a different single point of failure within this architecture.

Privacy Privacy is not addressed beyond SAML by this solution.

3.4 Decentralized solutions

Decentralized identity solutions do not rely on a single authority, but can defer to multiple. Many decentralized solutions, can also be classified as self-sovereign solutions, meaning that the user themselves fully own and manage their own data, without intervention from external administrators[16]. Often in a user-centric solution, this can take the form of a wallet containing credentials which prove their identity with the authority of different issuers or identity providers. This can be compared to having an ID or social security card in your own wallet, which can then be validated if need be, but most of the time as long as the issuer is considered trust-worthy further look-ups is not required [45]. One of the most commonly used technologies in modern decentralized solutions is blockchain, as can be seen in the spike of literature in decentralized solutions rising at around the same time blockchain was really asserting its position as a relevant technology. There are also recent attempts at decentralized solutions which do not rely on it, like *I Reveal My Attributes* (IRMA) from the Netherlands [45].

3.4.1 BIDaaS

The example in Figure 6 shows the separation between authentication of identity and attribute desimination in the approach proposed by BIDaaS [29]. The BIDaaS would operate on a private blockchain where "partners" of the BIDaaS would minimally be able to extract the user's virtual ID from the blockchain and verify it against the ID the user provides. Even though this paper matched the search keyword *federated identity management*, we consider it a decentralized proposal, since the identity controlled by the user through the ownership of a private key and can be used anywhere. However, this does not hold for the centrally stored attributes.

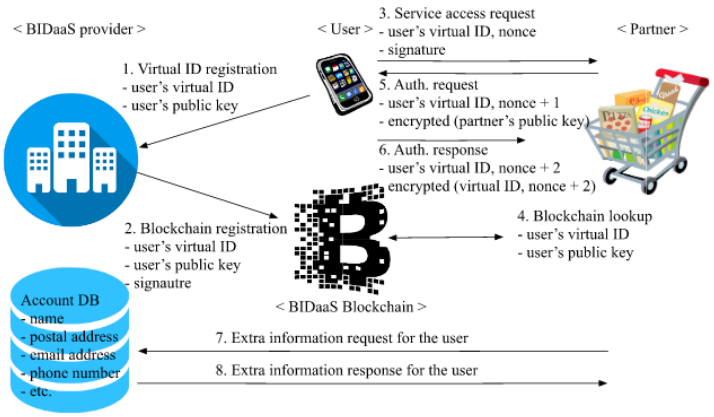


Figure 6: Example use-case from Lee [29]

Portability Portability is guaranteed for the identity beside any attributes stored, since knowledge of the private key by the user is sufficient to create the exact same identity at any other instance. In the example from Lee [29] attributes are stored in an external database.

Unlinkability User can create multiple identities for multiple service providers which should be an effective measure. If only one identity is created, it is trivial for any two service providers to perform identity linking. The identity provider is not involved in the authentication process, however the request of user attributes reveals the service provider that the user interacts with.

Loose coupling to service provider While the paper states that a service-level agreement between service provider and idenity provider must be established, in practice the service provider does not need to interact with the identity provider for authentication. If the blockchain is operated independently of this service, only the trust in the integrity of the identity providers private key is required.

Privacy The paper acknowledges that, in the currently proposed solution, there is no mechanism which restricts the service providers use of the user attributes in either time or scope of exploitation.

3.4.2 ShoCard

The ShoCard [42] solution, visualized in Figure 7, uses the Bitcoin blockchain to store the digest of an identification document which is held by the user and which is used as the digital identity. Attributes are attached to this identity as new Bitcoin transactions which point to the transaction containing the user's identity. A user can prove their identity by providing original data that produced the identity and certificate digest and by proving the ownership of the corresponding private key. Encrypted certificates can be backed up on a centralized server. This solution is intended for authentication with both physical and digital services.

Portability Portability does not apply to this solution, since all parties use the main Bitcoin network. Multiple centralized ShoCard servers could exist, since they are only required to store encrypted

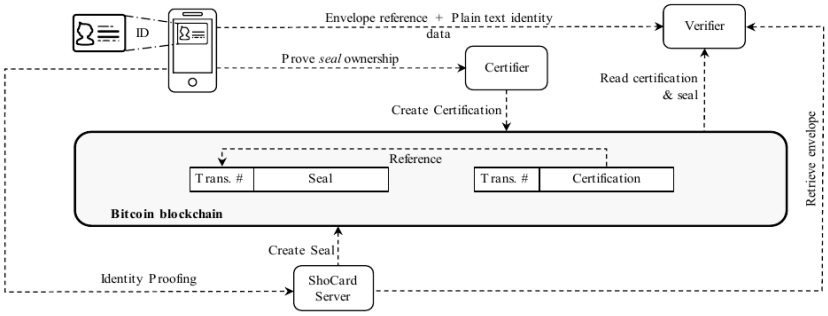


Figure 7: ShoCard components and interactions [18]

documents and interact with the blockchain. This functionality could even be provided by a service that is operated by the user, since it is not involved in any trust relation with the service providers.

Unlinkability Since certificates are only a hashed reference to the ShoCard identity, it is not possible to extract all certificates for any given identity from the blockchain. However, since users are expected to set up their ShoCard identity using their 'real-world' identity, the use of multiple identities does not seem to be an intendended feature and service providers could thus combine their knowledge on a specific user.

Loose coupling to service provider To verify a user's identity, a service provider would theoretically only need to interact with the user and the Bitcoin blockchain. The user could indicate the concrete ShoCard server at request-time. In order to verify a certificate (attributes), the service provider must trust the corresponding certifier.

Privacy While information is always encrypted outside of the context of a verification, the user's identity is always tied to the identity document that were used to set up the ShoCard such as the user's passport as noted by Dunphy and Petitcolas. This can lead to oversharing of personal information.

3.4.3 uPort

uPort[2] implements Decentralized Identifiers (DIDs) [24, 40] on the Ethereum blockchain which is implemented using smart contracts. This means that the whole ethereum network becomes the corresponding identity provider. Attributes are stored on the distributed storage service IPFS [6] and referenced on the blockchain by the user.

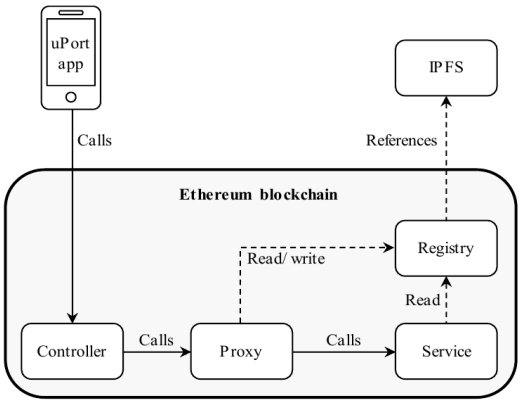


Figure 8: uPort architecture [18]

[2]In 2021 the project evolved into Veramo as announcement by the uPort project: `https://medium.com/uport/veramo-uports-open-source-evolution-d85fa463db1f` (Accessed June 3, 2022)

Portability Portability does not apply here if it can be assumed that the mainnet remains the only relevant Ethereum network.

Unlinkability Since the user can create an arbitrary number of distinct key pairs which correspond to unique identities, the threat of identity linking can be avoided.

Loose coupling to service provider Due to the elimination of all individual identity providers, the question of loose or tight coupling of a service provider to an identity provider does not apply. Instead, service providers interact with the blockchain. This solution does not address attribute validation which could be implemented similarly as in Sections 3.3.2 or 3.4.2.

Privacy Content which is stored on the blockchain cannot be removed and in the case of IPFS cannot be guaranteed to be removed, due to the nature of the append only leger and distributed storage service. This may have severe privacy implications if users attach sensitive data to their uPort identity and lead to accidental over-sharing [18].

3.4.4 Sovrin

Unlike the previously mentioned solutions that utilize decentralized ledger technology, Sovrin [43] explicitly uses a permissioned blockchain. Permissioned nodes are operated by trusted organizations. Users do not directly interact with the blockchain, but instead with an agent which could be self-hosted or provided by a third party. Similar to ShoCard, agents can also be used as an off-chain attribute store.

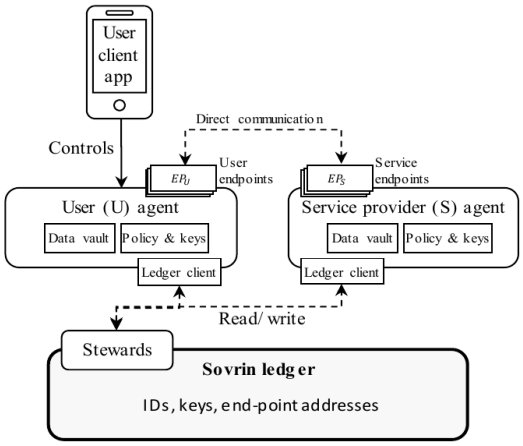


Figure 9: Sovrin architecture [18]

Portability Portability only applies with respect to the agents, because only the Soverin mainnet is intended to be used. Since users have the option of self-hosting agents or storing attributes locally, portability can be achieved.

Unlinkability Since the user can create an arbitrary number of distinct key pairs which correspond to unique identities, the threat of identity linking can be avoided.

Loose coupling to service provider Due to the elimination of all individual identity providers, the question of loose or tight coupling of a service provider to an identity provider does not apply. Instead, service providers interact with the blockchain and the user's agent which is standardized.

Privacy Through the agents, users are in full control of their information and the dissemination thereof. When using an agent provided by a third party, a trust relation between the provider of the agent and the user is required. The agent could also learn additional information, since it can interact with service providers on behalf of the user.

4 Solutions and their properties

While we can describe a lot of the solutions by the way they handle the previously discussed criteria, *portability*, *unlinkability*, *de-coupling* and *privacy*, we also wish to compare these solutions directly to see if certain approaches are generally more advantageous. Therefore, we compile them all in Table 4 the legend for which is given in Table 3. Both federated and decentralized approaches are evaluated on the same criteria together to see if general observations about those two groups can be made. However, it is important to understand that many services, by design, will not tackle some of these issues as well as others and instead prioritize different aspects of a modern identity management solution. This means that the solution, with the overall highest rating across all categories, is not automatically the most likely to establish itself as a new standard.
After showcasing the solution and how well they address our criteria, we will discuss common approaches and contrast noteworthy differences between solutions.

Score	Description
✓	Good handling of this criterion by the solution
(✓)	The solution addresses this criterion partially
✗	The solution does not adequately tackle the criterion
No symbol	This criterion does not apply to the solution

Table 3: How the solutions are rated

Solution	Type	Portability	Unlinkability	De-coupling	Privacy
BlindIdm	F	(✓)	(✓)	(✓)	✓
IC-Agents	F	✓	(✓)	✓	✓
Vo et al.	F		✓	(✓)	✓
Sh-IDaaS	F			(✓)	
BIDaas	D	✓	(✓)	✓	✗
ShoCard	D		(✓)	(✓)	(✓)
uPort	D		✓		✗
Sovrin	D	✓	✓		(✓)

Table 4: Comparison of (F)ederated and (D)ecentralized solutions

5 Discussion

For all decentralized blockchain based identity management solutions, we observe that privacy seems to present a challenge. Storing user data on the blockchain has severe privacy implications as we argue for uPort, however, even solutions such as BIDaaS or Sovrin that opt for an external storage have some disadvantages when compared to other solutions such as BlindIdm or Vo et al. which focus explicitly on the issue of privacy.
All the blockchain based decentralized solutions that we compared also significantly more complex to integrate or operate with for users and service providers while solutions like BlindIdm, Vo et al. or Sh-IDaaS which build on established federated identity management protocols such SAML or OpenID only add complexity to certain aspects of the authentication architecture.
Another important aspect that has to be considered for any blockchain based solution is the distributed ledger and corresponding consensus mechanism being used. Solutions like ShoCard or uPort which build on the Bitcoin or Ethereum mainnets respectively may be held back by highly limited transaction throughput resulting in long wait times and exorbitant fees which make adoption of such technologies for widespread and low-value services. Dedicated permissioned or permissionless networks on the other hand, such as the ones used by BIDaaS or Sovrin can use more efficient consensus mechanisms than Proof of Work and scale the network depending on the requirements of this singular use case. The use of community blockchains could lead to a paradigm shift where identity providers are replaced by a consortium operating a blockchain which runs a decentralized identity solution similar to uPort. Such a blockchain could be compared to today's Platform as a Service (PaaS) offerings.
IC-Agents seems the most promising approach to us, since it satisfies all our criteria, however,

they require extensive change to the current identity landscape which may even exceed that of the highlighted blockchain solutions. Additionally, the overhead posed by running an IC-Agent for every entity and through the use of onion routing for all user web traffic calls the practicality of this approach into question. Nevertheless, we are of the opinion that this approach has a lot of potential. Blockchain based solution, as well as others that are based on asymmetric cryptography, generally are more portable and potentially provider independent, since cryptographic challenges instead of trust relations between providers can be used to re-identify users. However, they also present new challenges such as key management, specifically in regard to invalidation and re-generation. While uPort and Sovrin each provide a mechanism for account recovery [18], key management is generally not yet very intuitive, as cryptocurrency wallet software demonstrates [33].

5.1 The struggles of emerging solutions

It is clear to us that the current identity management 'market' is dominated by the SSO services that Facebook, Google and Twitter offer, often having or currently using technologies or standards like SAML, OpenID or OAuth. This is despite the many efforts over recent years to continue to innovate in the area of identity management, as shown by the solutions presented in our survey. Despite this, most of the successful implementations are extensions of existing SSO-technologies, usually SAML. In 2018, it was observed that 6.3% of Alexa's top one million most visited pages supported SSO technology, where Facebook was at the time the largest identity provider [23].

The question that arises then is, why do so many services use the same established protocols [46]? One hypothesis that one quickly arrives at is that users are reluctant to pay someone to manage your identity for them. This amplifies the network effect experienced by established SSO providers in the consumer space, the vast majority of which are provided free of charge. Another advantage most of the well known SSO-solutions have is their ease-of-use and simplicity, often due to them being actively promoted by the corresponding ecosystems of the identity provider. Technological innovation, even with high improvements in security and privacy with significant enthusiasm from (potential) users, cannot completely counteract the overwhelming barrier of lacking usability. With mandatory components of more very highly demanded products or widespread adoption in a corporate identity management being perhaps an exception, only slow organic growth can be expected. Finally, professionals and researchers might overestimate the public's concern for privacy. While there might be some outcry over violations regarding the usage of personal information, convenience seems to be detrimental to the effort put into solving the issues by the consumers. In Europe, expectations may be placed in legislation pushing for change and government initiatives such as citizen identity services under eID [4]. However, as the emergence of unintuitive cookie pop-ups on websites in response to the General Data Protection Regulation (GDPR) has shown, unspecific legislative action may produce unfavorable outcomes [44].

5.2 Observed trends and prediction of future developments

Among the reviewed approaches, three (BlindIdm, Vo et al., Sh-IDaaS) are built on well established federated identity protocols and support SAML. This shows that there is still interest in this technology and new commercial solutions may want to leverage these existing standards instead of establishing competing similar ones.
With the rise of cryptocurrencies, the concept of self-sovereign identity has seen new developments. We have reviewed four different solutions (BIDaaS, ShoCard, uPort, Sovrin) which utilize different blockchains and to a different extent. This shows that the emergence of this technology did not yield an obvious solution to this challenge. Using asymmetric cryptography is not exclusive to blockchain, as the proposal for IC-Agents shows. Nevertheless, the momentum behind this technology will likely continue to spawn new innovation, such as the Unified Identity Protocol (UID) proposed by IOTA [32], which may be refined and scalable enough to be widely deployed.

6 Future work

We limited our comparison to eight federated and decentralized identity management solutions, however there are several other publications, such as Millenaar and Yarger [32], Zhaofeng et al. [51], Alom et al. [5], and Yildiz et al. [50], which could be compared using our criteria.
Since some approaches are only very specific about certain aspects of the proposed identity, it may be possible to combine multiple approaches, such as the decentralized attribute store of uPort with the privacy preserving proxy re-encryption of BlindIdm. These should be evaluated using the same criteria to determine which combinations present an improvement over the original approaches.
Our survey focused on general identity management services, however, different solutions may have unique qualities and intended purposes. A use-case focused study may provide more relevant suggestions for different kinds of services such as internet of things and mobile applications, commercial end-user, corporate or public/government services.

7 Conclusions

In this survey, we reviewed whether recent developments regarding identity management in the cloud and emerging technologies can be an indicator of the future of identity management. This was achieved by conducting a literature survey concerning publication trends over time and in relation with each other. Next, isolated identity services are introduced that are then compared with two alternative general approaches, which are federated identity services and decentralized identity management. We also try to determine which approaches are the most promising by diving deeper into a selection of notable solutions from each approach. These solutions are reviewed and compared based on their impact on four criteria: *portability*, *unlinkability*, *de-coupling* and *privacy*. We found that no one single solutions of the ones that we investigate satisfies all criteria without some drawbacks. The blockchain specifically bare some privacy risks. Finally, we discuss some potential challenges faced by federated and decentralized approaches, as well as any new identity management solution. We see wide scale adoption as a key challenge to any solution. We conclude that established federated protocols such as SAML will remain relevant while decentralized approaches continue to innovate.

References

[1] Semantic Scholar API. URL https://www.semanticscholar.org/product/api. (Accessed June 3, 2022).

[2] Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0, 2005. URL http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf. (Accessed June 3, 2022).

[3] OpenID Authentication 2.0 - Final, 2007. URL https://openid.net/specs/openid-authentication-2_0.html. (Accessed June 3, 2022).

[4] Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. *OJ*, L 257/73:73—-114, 2014.

[5] Ifteher Alom, Romana Mahjabin Eshita, Anam Ibna Harun, Md Sadek Ferdous, Mirza Kamrul Bashar Shuhan, Mohammad Jabed M Chowdhury, and Mohammad Shahidur Rahman. Dynamic Management of Identity Federations using Blockchain. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2021. doi: 10.1109/ICBC51069.2021.9461128.

[6] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System, 2014. URL https://arxiv.org/abs/1407.3561.

[7] Elisa Bertino, Federica Paci, Rodolfo Ferrini, and Ning Shang. Privacy-preserving digital identity management for cloud computing. *IEEE Data Eng. Bull.*, 32(1):21–27, 2009.

[8] Eleanor Birrell and Fred Schneider. Federated identity management systems: A privacy-based characterization. *Security & Privacy, IEEE*, 11:36–48, 09 2013. doi: 10.1109/MSP.2013.114.

[9] Dan Blum. Control Access with Minimal Drag on the Business. In *Rational Cybersecurity for Business*, pages 227–257. Apress, 2020. doi: 10.1007/978-1-4842-5952-8_8. URL https://doi.org/10.1007/978-1-4842-5952-8_8.

[10] Miguel A. Calles. *Authentication and Authorization*, pages 229–256. Apress, Berkeley, CA, 2020. ISBN 978-1-4842-6100-2. doi: 10.1007/978-1-4842-6100-2_9. URL https://doi.org/10.1007/978-1-4842-6100-2_9.

[11] Jan Camenisch, Abhi Shelat, Dieter Sommer, Simone Fischer-Hübner, Marit Hansen, Henry Krasemann, Gérard Lacoste, Ronald Leenes, and Jimmy Tseng. Privacy and identity management for everyone. In *Proceedings of the 2005 workshop on Digital identity management*, pages 20–27, 2005.

[12] Jan Camenisch, Ronald Leenes, and Dieter Sommer. *Digital Privacy: PRIME-Privacy and Identity Management for Europe*, volume 6545. Springer, 2011.

[13] Scott Cantor, Steven Carmody, Marlena Erdos, Keith Hazelton, Walter Hoehn, RL "Bob" Morgan, Tom Scavo, and David WasleyS. Shibboleth Architecture - Protocols and Profiles, 2005. URL https://shibboleth.net/documents/internet2-mace-shibboleth-arch-protocols-200509.pdf. (Accessed June 3, 2022).

[14] Jan De Clercq. Single Sign-On Architectures. In *Infrastructure Security*, pages 40–58. Springer Berlin Heidelberg, 2002. doi: 10.1007/3-540-45831-x_4. URL https://doi.org/10.1007/3-540-45831-x_4.

[15] Damiano Di Francesco Maesa and Paolo Mori. Blockchain 3.0 applications survey. *Journal of Parallel and Distributed Computing*, 138:99–114, 2020. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2019.12.019. URL https://www.sciencedirect.com/science/article/pii/S0743731519308664.

[16] Omar Dib and Khalifa Toumi. Decentralized identity systems: Architecture, challenges, solutions and future directions. *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, pages 2516–0281, 2020.

[17] Gabi Dreo, Mario Golling, Wolfgang Hommel, and Frank Tietze. ICEMAN: An architecture for secure federated inter-cloud identity management. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1207–1210, 2013.

[18] Paul Dunphy and Fabien A.P. Petitcolas. A first look at identity management schemes on the blockchain. *IEEE Security Privacy*, 16(4):20–29, 2018. doi: 10.1109/MSP.2018.3111247.

[19] Christian Emig, Frank Brandt, Sebastian Kreuzer, and Sebastian Abeck. Identity as a service – towards a service-oriented identity management architecture. In *Dependable and Adaptable Networks and Services*, pages 1–8. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-73530-4_1. URL https://doi.org/10.1007/978-3-540-73530-4_1.

[20] J. Fayolle, M. Ates, S. Ravet, and A. Ahmat. An identity-centric internet: Identity in the cloud, identity as a service and other delights. In *2012 Seventh International Conference on Availability, Reliability and Security*, pages 555–560, Los Alamitos, CA, USA, aug 2011. IEEE Computer Society. doi: 10.1109/ARES.2011.85. URL https://doi.ieeecomputersociety.org/10.1109/ARES.2011.85.

[21] Serge Feldman. Building a Better Search Engine for Semantic Scholar (Blog), 2020. URL https://blog.allenai.org/building-a-better-search-engine-for-semantic-scholar-ea23a0b661e7. (Accessed June 3, 2022).

[22] Jason Garman. *Kerberos: The Definitive Guide*. O'Reilly & Associates, Inc., USA, 2003. ISBN 0596004036.

[23] Mohammad Ghasemisharif, Amrutha Ramesh, Stephen Checkoway, Chris Kanich, and Jason Polakis. O Single {Sign-Off}, Where Art Thou? An Empirical Analysis of Single {Sign-On} Account Hijacking and Session Management on the Web. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1475–1492, 2018.

[24] Amy Guy, Manu Sporny, Drummond Reed, and Markus Sabadello. Decentralized identifiers (DIDs) v1.0. W3C proposed reccommendation, W3C, August 2021. https://www.w3.org/TR/2021/PR-did-core-20210803/.

[25] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012. URL https://www.rfc-editor.org/info/rfc6749. (Accessed June 3, 2022).

[26] Audun Jøsang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. In *Proceedings of the 2005 Australasian Workshop on Grid Computing and E-Research - Volume 44*, ACSW Frontiers '05, page 99–108, AUS, 2005. Australian Computer Society, Inc. ISBN 1920682260.

[27] Michael Kubach, Christian H Schunck, Rachelle Sellung, and Heiko Roßnagel. Self-sovereign and decentralized identity as the future of identity management? *Open Identity Summit 2020*, 2020.

[28] Maryline Laurent, Julie Denouël, Claire Levallois-Barth, and Patrick Waelbroeck. 1 - digital identity. In Maryline Laurent and Samia Bouzefrane, editors, *Digital Identity Management*, pages 1–45. Elsevier, 2015. ISBN 978-1-78548-004-1. doi: https://doi.org/10.1016/B978-1-78548-004-1.50001-8. URL https://www.sciencedirect.com/science/article/pii/B9781785480041500018.

[29] Jong-Hyouk Lee. Bidaas: Blockchain based id as a service. *IEEE Access*, 6:2274–2278, 2018.

[30] Mengyi Li, Chi-Hung Chi, Chen Ding, Raymond Wong, and Zhong She. A multi-protocol authentication shibboleth framework and implementation for identity federation. In Raheem Beyah, Bing Chang, Yingjiu Li, and Sencun Zhu, editors, *Security and Privacy in Communication Networks*, pages 81–101, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01704-0.

[31] John Mears. The rise and rise of ID as a service. *Biometric Technology Today*, 2018(2):5–8, February 2018. doi: 10.1016/s0969-4765(18)30023-7. URL https://doi.org/10.1016/s0969-4765(18)30023-7.

[32] Jelle Femmo Millenaar and Mathew Yarger. The Case for a Unified Identity – Our Vision for a Unified Identity Protocol on the Tangle for Things, Organizations, and Individuals (White Paper), 2019. URL https://files.iota.org/comms/IOTA_The_Case_for_a_Unified_Identity.pdf. (Accessed June 3, 2022).

[33] Md Moniruzzaman, Farida Chowdhury, and Md Sadek Ferdous. Examining Usability Issues in Blockchain-Based Cryptocurrency Wallets". In Touhid Bhuiyan, Md. Mostafijur Rahman, and Md. Asraf Ali, editors, *Cyber Security and Computer Science*, pages 631–643, Cham, 2020. Springer International Publishing. ISBN 978-3-030-52856-0.

[34] Omar Mubin, Fady Alnajjar, Abdullah Shamail, Suleman Shahid, and Simeon Simoff. The new norm: Computer Science conferences respond to COVID-19. *Scientometrics*, 126(2): 1813–1827, 2021. doi: 10.1007/s11192-020-03788-9. URL https://doi.org/10.1007/s11192-020-03788-9.

[35] B. Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 32:33 – 38, 10 1994. doi: 10.1109/35.312841.

[36] David Nuñez and Isaac Agudo. Blindidm: A privacy-preserving approach for identity management as a service. *International Journal of Information Security*, 13:199–215, 2014.

[37] David Nuñez, Isaac Agudo, and Javier López. 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*, pages 241–248, 2012.

[38] David Nuñez, Isaac Agudo, and Javier López. Leveraging privacy in identity management as a service through proxy re-encryption. In *ESOCC 2013*, 2013.

[39] David Nuñez, Isaac Agudo, and Javier Lopez. *Privacy-Preserving Identity Management as a Service*, pages 114–125. Springer International Publishing, Cham, 2015. ISBN 978-3-319-17199-9. doi: 10.1007/978-3-319-17199-9_5. URL https://doi.org/10.1007/978-3-319-17199-9_5.

[40] Andreea-Elena Panait, Ruxandra F. Olimid, and Alin Stefanescu. Analysis of uPort Open, an Identity Management Blockchain-Based Solution. In Stefanos Gritzalis, Edgar R. Weippl, Gabriele Kotsis, A. Min Tjoa, and Ismail Khalil, editors, *Trust, Privacy and Security in Digital Business*, pages 3–13, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58986-8.

[41] Sebastian Rieger. User-Centric Identity Management in Heterogeneous Federations. In *2009 Fourth International Conference on Internet and Web Applications and Services*. IEEE, 2009. doi: 10.1109/iciw.2009.85. URL https://doi.org/10.1109/iciw.2009.85.

[42] SITA; ShoCard.
Travel Identity of the Future (White Paper), 2016. URL https://blockchainlab.com/pdf/2016-05-00-idm-ShoCard-travel-identity-of-the-future.pdf. (Accessed June 3, 2022).

[43] Andrew Tobin, Drummond Reed, and Phillip J. Windley. The Inevitable Rise of Self-Sovereign Identity (White Paper), 2017. URL https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf. (Accessed June 3, 2022).

[44] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. (Un)Informed Consent: Studying GDPR Consent Notices in the Field. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 973–990, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354212. URL https://doi.org/10.1145/3319535.3354212.

[45] Dirk Van Bokkem, Rico Hageman, Gijs Koning, Luat Nguyen, and Naqib Zarin. Self-sovereign identity solutions: The necessity of blockchain technology. *arXiv preprint arXiv:1904.12816*, 2019.

[46] Anna Vapen, Niklas Carlsson, Anirban Mahanti, and Nahid Shahmehri. A Look at the Third-Party Identity Management Landscape. *IEEE Internet Computing*, 20(2):18–25, 2016. doi: 10.1109/MIC.2016.38.

[47] Tri Hoang Vo, Woldemar Fuhrmann, and Klaus-Peter Fischer-Hellmann. How to adapt authentication and authorization infrastructure of applications for the cloud. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 54–61, 2017. doi: 10.1109/FiCloud.2017.14.

[48] Tri Hoang Vo, Woldemar Fuhrmann, and Klaus-Peter Fischer-Hellmann. Privacy-preserving user identity in identity-as-a-service. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–8, 2018. doi: 10.1109/ICIN.2018.8401613.

[49] Tri Hoang Vo, Woldemar Fuhrmann, Klaus-Peter Fischer-Hellmann, and Steven Furnell. Identity-as-a-service: An adaptive security infrastructure and privacy-preserving user identity for the cloud environment. *Future Internet*, 11(5), 2019. ISSN 1999-5903. doi: 10.3390/fi11050116. URL `https://www.mdpi.com/1999-5903/11/5/116`.

[50] Hakan Yildiz, Christopher Ritter, Lan Thao Nguyen, Berit Frech, Maria Mora Martinez, and Axel Küpper. Connecting Self-Sovereign Identity with Federated and User-centric Identities via SAML Integration. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2021. doi: 10.1109/ISCC53001.2021.9631453.

[51] Ma Zhaofeng, Meng Jialin, Wang Jihui, and Shan Zhiguang. Blockchain-Based Decentralized Authentication Modeling Scheme in Edge and IoT Environment. *IEEE Internet of Things Journal*, 8(4):2116–2123, 2021. doi: 10.1109/JIOT.2020.3037733.

HPC and Cloud (Potentials, Challenges and Limits)

Muhammad Hasseeb Qutabzada
Masters in Big Data Engineering (13879421)
University of Amsterdam
Vrije University
hasseeb.qutabzada@student.uva.nl

Li Mingchen
Masters in Computer Science (14116006)
University of Amsterdam
mingchen.li@student.uva.nl

June 3, 2022

Abstract

In this article, keeping in view the ever increasing demands of high performance computing, the potential solutions offered by Cloud are examined. Although, interest for Cloud Computing shown by HPC users is gaining pace but there exists many challenges and limits. In previous works, the relationship between HPC and Cloud is analysed on single aspects. The aim of this study is to present a holistic analysis of the potentials, challenges and limits when it comes to the relationship between High Performance Computing and Cloud. It evaluates this relationship in terms of performance of HPC applications in the Cloud, cost-comparison between HPC and Cloud, environmental impact of HPC in Cloud, energy optimisation, hybrid computing and fault tolerance for HPC in Cloud.

1 Introduction

High Performance Computing is the capability to process data and perform calculations at high speeds. Supercomputers that make use of thousands of compute nodes to process tasks are an example of HPC solutions. Advancements in science, society and industry have HPC as their backbone as the role of data has become crucial. With the rapid rise in Internet of Things and Artificial Intelligence, an exponential increase in the size of data is followed. This data requires to be processed at, high speeds, which are a key feature of HPC solutions. The architecture of HPC comprises compute, network and storage resources.

These components are combined to establish an HPC cluster consisting of thousands of compute servers which are networked together. HPC has uses in research labs, oil and gas, media and entertainment, artificial intelligence, machine learning, financial services, medical industry, innovation.

Cloud Computing delivers computing services that include servers, network, storage, software, analytics, intelligence etc over the internet to provide flexible resources, economies of scale and high-paced innovation. The definition provided by [CC and Grid Computing 360] covers the concept comprehensively:

A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet. Cloud Computing is gaining popularity across public and private organisations as it solves their ever growing demands of computing and storage. Factors such as low hardware costs, better compute and storage resources, exponential rise in the size of data, widespread adoption of web services have favoured the rise of Cloud Computing. The architecture in a Cloud has three main layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Considering the limitations posed by traditional HPC platforms, Cloud Computing being resourceful in computational power is offering possible solutions for the users of HPC. Tuning the computing platform as per the application needs and budget of users, Clouds provide a cost-effective and timely solution. As opposed to the lengthy procedures of setting up a HPC platform and expensive maintenance, Cloud services could be obtained in minutes. Elastic nature of resources, ability to offer infrastructure and software and virtualisation are the key features of Cloud Computing that have motivated HPC users to consider Cloud as a potential solution. Although, discovery of potential in Cloud Computing for HPC has been a driving force in consideration of porting HPC applications to Cloud but there exist many challenges and limits. In previous work, considering performance as a metric the Cloud's ability to handle HPC applications [ImprovingHPCperformance], results have not been promising. Cloud had been unable to meet the demands of network performance required by HPC applications and other limitations occurred due to resource heterogeneity and multi-tenancy. Nonetheless, recent advancements in the Cloud have been aimed at improving the technology to address these issues. In previous work, the researches have focused on single aspects of the relationship between Cloud Computing and HPC. In this article, we aim to critically analyse the potential solutions offered by Cloud Computing for HPC users and illustrate the challenges and limits that exist in adoption of Cloud by HPC users. This article covers the relationship between Cloud and HPC from all aspects including improving performance of HPC applications on Cloud, cost-comparison, environmental impact, energy optimisation, hybrid computing, fault-tolerance approach for HPC in Cloud, prediction of Cloud performance for HPC applications. The rest of this article is divided into four further sections: (2) Body, (3) Discussion, (4) Conclusions and (5) References. In the next section, these various aspects of HPC and Cloud are analysed in terms of potentials, challenges and limits.

2 Body

In this section, relationship between HPC and Cloud is examined through various aspects to draw a meaningful analysis of the potentials that exist, together with highlighting the challenges and limits.

2.1 Performance of HPC applications on Cloud

As there are several cloud platforms, the performance of HPC applications depends upon the type of Cloud used. To demonstrate that, three public clouds were technically evaluated for their capability and suitability to run HPC applications [1]. Belonging to the IaaS (Infrastucture-As-A-Service) type, these clouds provide the researchers with full control of a Virtual Machine for application installation and customisation of computing resources. The experimentation techniques involve using benchmarks that are then deployed to three public clouds.

The benchmarks used include NPB (NAS Parallel Benchmark), HPL (High Performance LINPAC), CSFV (Cubed Sphere Finite Volume). Clouds to which these benchmarks are deployed are Amazon EC2 Cloud, GoGrid Cloud and IBM Cloud. Inter-process communication over the network was not considered when evaluating the single cloud server instance performance. Focusing on virtualisation overhead, NPB-OMP was run on these clouds and their performance was compared to NCSA supercomputer. Figure 1 shows that the performance of all three clouds was comparable to NCSA supercomputer. This comparable performance shows that virtualisation does not have a significant effect on the overhead. IBM turned out to be the best Cloud owing to its latest processor technology at that time.

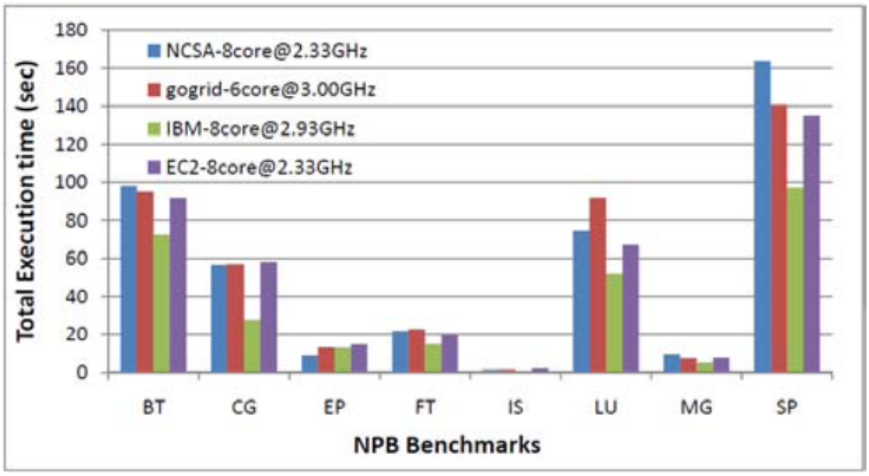


Figure 1: NPB OMP on public clouds [1]

It is interesting to note that by running NPB-MPI benchmark on three clouds and their comparison with NCSA, IBM Cloud turned out to have the worst performance despite its latest technology. This throws light on the role of networking in MPI based parallel computing. Although, EC2 and GoGrid seem to have similar performance, the use of 6 core-per-node server [1] configuration in GoGrid leads to more inter-process communications and thereby, GoGrid is expected to give better results if 8 core-per-node server is used.

HPL benchmarking in GoGrid brings some optimism as increasing cluster from 1 to 5, the GFLOPS reach to 165 from 55. On the other hand, in case of EC2 and IBM, the HPL benchmarking does not advocate the idea of running HPC applications. CSFV benchmarking also holds GoGrid as the best performer out of the three clouds, with IBM performing the worst and having scaling problems. It has been shown GoGrid does have the potential of underperforming NASA system [1] by only 10-20 percent if oversubscription happens. These three clouds have different performances when HPC applications are ran, but the paper [1] doesn't suggest a cloud out of these three as the best one. The challenges and limits vary as the GoGrid can improve if it opens up 8 or more cores per node, IBM can improve by upgrading their network with Gigabit NICs and switches. A limitation of this research is the cloud cluster size being 10 nodes approximately.

As Amazon's EC2 Cloud seemed to underperform either IBM Cloud or GoGrid depending upon the benchmarks deployed [1], a relatively new study [7] suggests a new approach to improve HPC applications performance and scalability on EC2. Since sorting algorithms tend to be a crucial part of HPC applications, this study [7] implements MPI version of parallel Radix sort on EC2 and high-end HPC system i.e Sirius and then does a comparative analysis. MPI was selected as it outperforms others when it comes to parallel sorting. To explore the scalability in relation to data size, execution time and speed ups are calculated for four workloads of 25, 50, 75 and 100MB by scaling up the nodes from 2, 4, 6 to 8 nodes. Figure 2 show the comparisons for 25MB and 100MB data and Figure 3 shows how different workloads performed with 2 nodes and 8 nodes. Data distribution has an impact and uneven distribution of data poses a challenge as it causes a delay in execution time. Maximum speedup was observed from 2 to 4 nodes, however further increasing the nodes resulted in performance degradation. However, this study [7] finds EC2 to be outperforming dedicated HPC cluster Sirius in all cases. By comparing the performance of EC2 in [1] and [7], we realise that advancement in Cloud is enlarging the potential for HPC users and EC2's suitability for HPC has increased, however, EC2 has limitations and is considered acceptable for on-demand and small sized HPC applications [7].

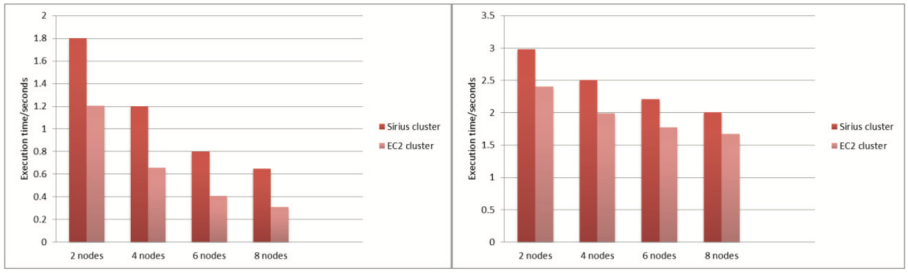


Figure 2: Average execution times (sec) for 25MB (left) and for 100MB (right) [7]

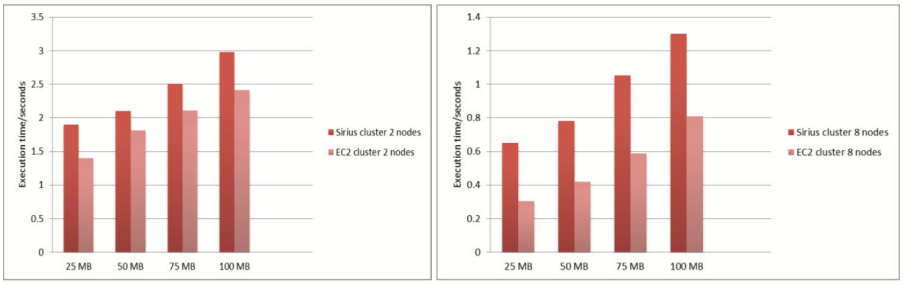


Figure 3: Computation performance with 2 nodes (left) and 8 nodes (right) [7]

2.2 Cost-comparison of HPC and Cloud

Evaluation of Cloud platforms for HPC applications has mostly focused on the performance metric, giving less attention to the cost comparisons. In this article, we examine a study [11] that focuses on cost metric by comparing Amazon EC2 to an in-house HPC system. The study [11] notes varying prices of Cloud platforms and constant evolution of Cloud, bias in the direct price comparison of HPC operating cost to a Cloud instance as this comparison leaves out the performance metric as the gaps in academic research and thereby, proposes a new cost analysis model. The comparison is threefold. Firstly, it goes for a Total Cost of Ownership (TCO) analysis of the HPC facility at University of Luxembourg which considers all costs that are incurred. TCO is decomposed into two major types of costs CAPEX and OPEX [11]. The TCO analysis in combination with correct amortisation period of items in TCO, lead to a node hourly cost. As the usage of HPC tends to play a part in TCO analysis, the study [11] used the HPC facility by University of Luxembourg where the access for constant monitoring of HPC usage was undertaken.

The computing performance metric used to compare the EC2 instances and HPC computing nodes, is the GFLOPS. Performances of EC2 instances is given in ECU (EC2 Compute Units), but the lack of information on how ECU is computed and the presence of a linear relationship between ECU and GFLOPS,

have caused the researchers to use GFLOPS as the performance metric. A multiple linear regression approach is employed to compute the Amazon EC2 price model, while keeping in consideration the different purchasing modes. Figure 4 shows a complete cost analysis, where only in a few cases, EC2 instances are cost effective. Otherwise, it is always more cost effective to acquire the resources for HPC facility. It should however be noted that this cost comparison is carried out by considering the HPC facility of University of Luxembourg.

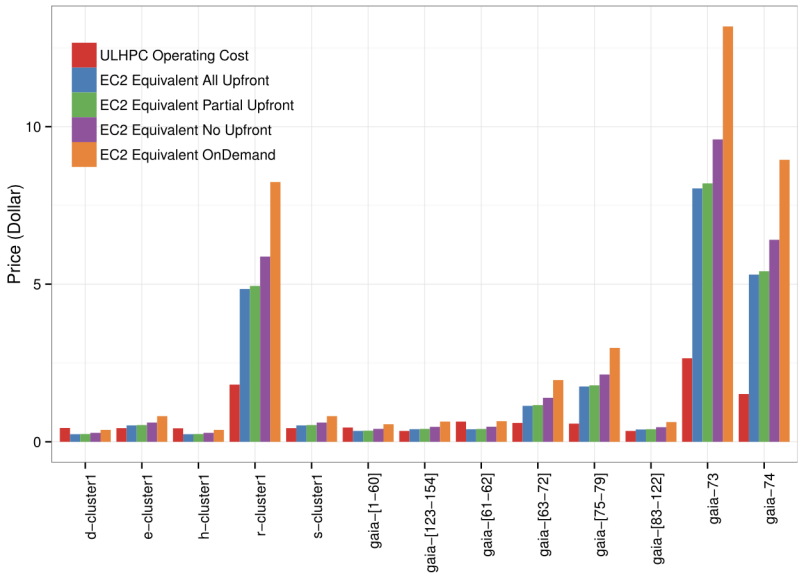


Figure 4: Comparison of in-house HPC operating cost to Amazaon EC2 for different purchasing modes [11]

Another cost comparison is carried out by comparing Purdue University's HPC community cluster program to a commercial cloud [9]. The strategy deployed actually presents a per node-hour cloud computing cost that depends on the actual usage of community cluster users. Owned by the faculty and centrally owned by the institution, a community cluster generally benefits research groups. Ran by their designated technical staff, low overheads due to centralisation, pooling of resources by all participants and meaningful sourcing of the cluster hardware make community clusters the best option for faculty when it comes to an in-house facility. Purdue operates two large community clusters, namely, Steele and Coates.

A instance of Amazon's EC2 called "Cluster Compute Instances" (CCI) was directed as an alternative to traditional cluster-based computing. The study [9] analyses this EC2 CCI product from performance and cost perspectives. For comparison of performance, NPB benchmarks are deployed. From figure 5, we can learn the results which show EC2 to be much faster than Coates or Steele. Nonetheless, this performance is only valid for CPU bound HPC applications. For

cost analysis, a methodology is applied to consider all the factors that add to the costs of community cluster and EC2 CCI instance and EC2 CCI equivalent. In cost analysis, the community cluster tends to be the best choice and more affordable for the faculty than providing nodes in Amazon EC2. The Purdue owner has to pay $1.75 to get the same computation done that is achieved by Purdue cluster for 0.25$ only. The setup of EC2 CCI could be difficult and challenging for end-users, however, it has the potential to be useful in situations where time constraints are present. To be cost effective, there lies a potential for EC2 CCI, as they could provide "spot instances" with prices fluctuating in an open market.

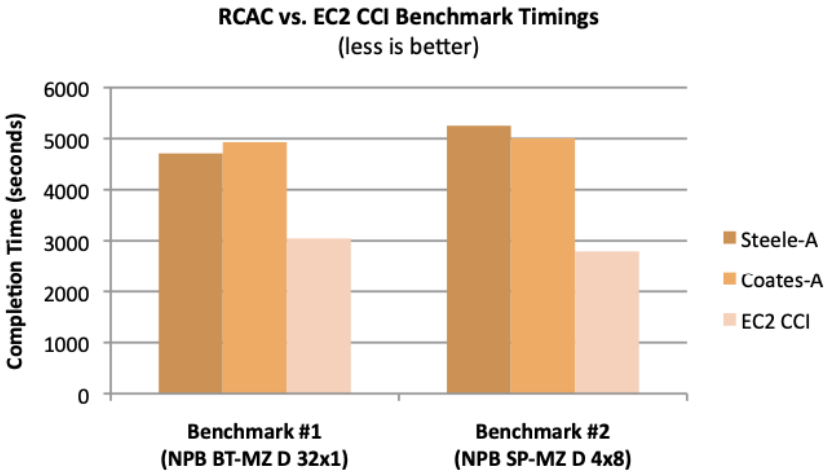


Figure 5: NPB performance on Purdue clusters and EC2 CCI [9]

2.3 Environmental Impact of HPC in Cloud and Energy Optimisation

As environmental impact of any practice does seem to shape and influence its future, in this article, we review techniques that could make running HPC applications in Cloud more environmental friendly. We shall critically analyse a study [15] that focuses on how the Cloud provider can achieve optimal energy sustainability for HPC applications by taking advantage of the heterogeneity of multiple data centres in different parts of the world. In this article, we present an analysis of this study that aims to lower the carbon footprint without compromising on the profits of Cloud providers.

A meta-scheduling model has been proposed which follows a protocol and acts as an interface to the Cloud. This meta-scheduling protocol can be seen in Figure 6. It basically has two phases. The first one is mapping phase where an application is mapped onto a data centre. Second is scheduling

phase where applications are scheduled in the data centre that was chosen for mapping. For optimising the global carbon emission these policies are employed: Greedy Minimum Carbon Emission, Minimum-Carbon-Emission-Minimum-Carbon-Emission. While, for optimising the global profit of all data centres, the following policies are in place: Greedy Maximum Profit, Maximum-Profit-Maximum-Profit. To strike a balance between carbon emissions and profits, the MCE-MP (Minimising carbon emission and maximising profit) is used. The policies employed [15] for scheduling have shown an impressive saving of 25 percent in energy when compared to profit-based scheduling policies.

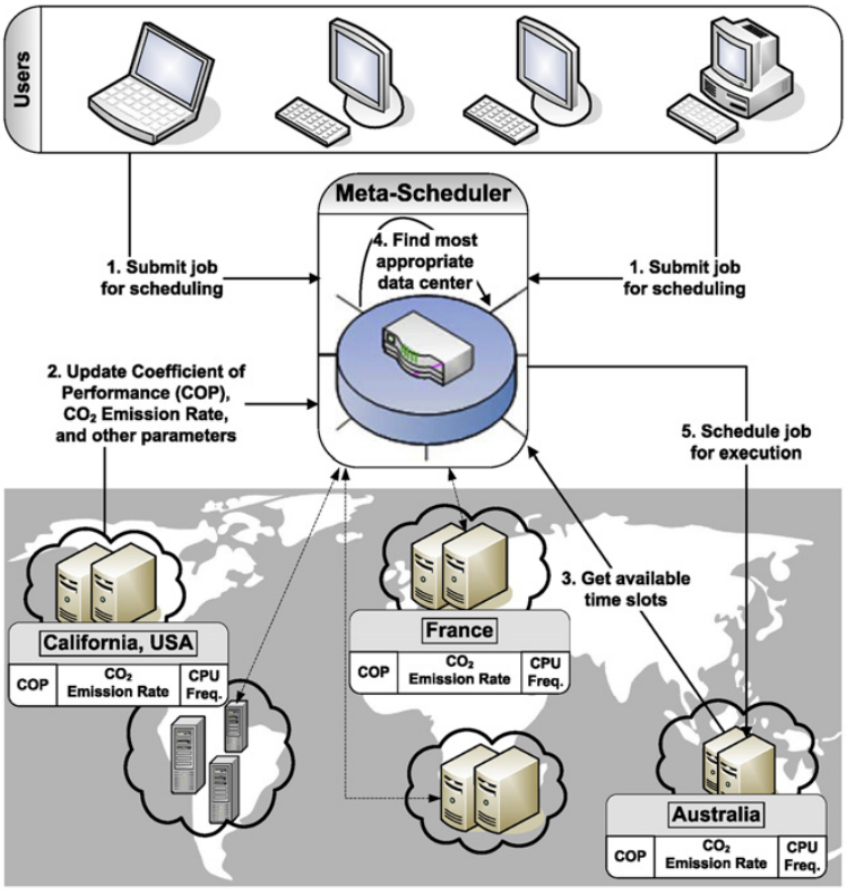


Figure 6: Cloud meta-scheduling protocol [15]

This article also analyses an approach that presents HPC based clouds for energy optimisation [10]. Objective behind this approach is reduction of expenses in task processing in terms of energy. Two models have been developed [10] for use as an HPC based cloud: CometCloud based system and HTCondor based

system. While, workers are coordinate by Master and thus carry out tasks in CometCloud, for HTCondor there exists a Master-Slave Mechanism.

A simulation software i.e EnergyPlus plays a role in this approach as researchers have shown how HPC based cloud can be used for deploying EnergyPlus simulation-based optimisation. The evaluation of optimisation process is carried out from two sides: Facility manager perspective and HPC cloud provider perspective. The former represents the users who are interested to minimise energy costs, while the later is represented by any data centre that has the ability to process EnergyPlus based simulations. Understanding the fact that an optimisation problem relies on several runs of EnergyPlus, the researches have identified two important parameters: 1) Complexity of the Input Data File building model and 2) Period to Simulate. The complexity of the input data file has a direct impact on total execution time and simpler input files require less time. Period to simulate impacts simulation time and simulation of building model over shorter period requires less time to simulate than a building model over a long period.

For this real-time energy optimisation, some more limits and challenges exist. Two other parameters must be complied with by the HPC based cloud and these are: Time-To-Complete and Results Quality.

The two environments namely CometCloud and HTCondor that are integrated with HPC based cloud in [10]. CometCloud being an autonomic computing engine basing upon Comet decentralised coordination substrate, supports heterogeneity and dynamic infrastructures of cloud or grid or HPC, allows for integration of public and private clouds. While, HTCondor is an high throughput work management software for cluster of computing resources. To enhance the performance of distributed computing resources, wide application of HTCondor can be noticed. For those interested in the architecture of these two environments, Figure 7 seems to be the answer.

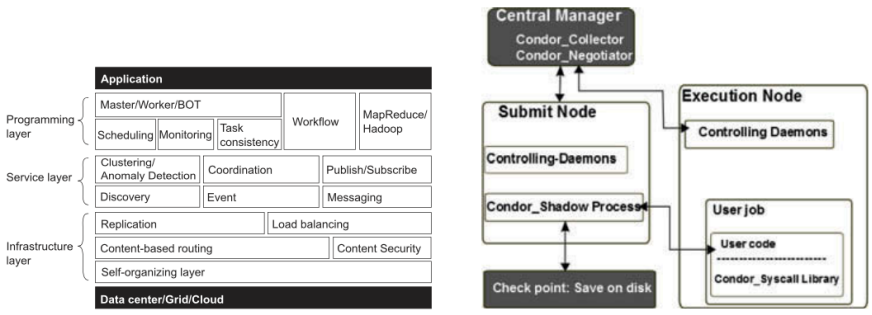


Figure 7: CometCloud architecture (left) and HTCondor architecture (left)

For experimentation, three environments are used: 1) Single CometCloud — local 2) Federated CometCloud — shared within a federation system and 3) HTCondor. The types of jobs and the workloads of jobs has been changed in the

experiments [10]. A number of different experiments are carried out that observe time with job execution, tasks completed in deadline, data transferred with job execution, data storage with job execution, total cost with job execution and so on. The benefits of HPC based cloud lie in the areas of task completion and cost optimisation. More tasks result in more cost and the major increase is attributed to cost of storage and execution. So, these expensive operations of storage and execution present a challenge for HPC based clouds. Nonetheless, the cost with transfer turns out to be linear over time. From an overall perspective, the idea of HPC based cloud is termed beneficial by [10] and we examine it in this article as this solution could be extended to other type of problems. This approach combines the performances of HPC system with a unique combination of different environments such as CometCloud and HTCondor and thus, it reaches an optimum between execution time, costs and scheduling.

2.4 Hybrid Computing

The benefits of High Performance Computing, Cloud Computing and Grid Computing are remarkable in their respective domains. In this article, we analyse a study [5] that introduces a new hybrid approach for HPC that predicts execution of applications, scales well from a resource to multiple resources having different owners, location and policies. For this novel hybrid approach, an architecture is suggested that aims to combine the benefits of these three technologies. The foundation of this architecture lies in the block named Elastic Cluster. In addition to accruing the benefits of this Elastic Cluster in predicting the execution of HPC workloads, a new distributed information system is also proposed which merges features of distributed hash tables and relational databases.

As hybrid computing is the amalgamation of three different technologies, each technology has characteristic attributes namely: 1) resource ownership, 2) resource accessibility, 3) resource sizing, 4) resource allocation policy and 5) application portability. The study [5] well describes HPC facility, Cloud Computing and Grid Computing in terms of these attributes. Figure 8 shows comparison of these three paradigms.

Attribute	HPC	Grid	Cloud
Capacity	fixed	average to high; growth by aggregating independently managed resources	**high; growth by elasticity of commonly managed resources**
Capability	**very high**	average to high	low to average
Virtual Machine Support	rarely	sometimes	**always**
Resource sharing	limited	**high**	limited
Resource heterogeneity	low	**average to high**	low to average
Built-in Workload Management	**yes**	**yes**	no
Distribute Workload Across Resources from Multiple Admin Domains	not applicable	**yes**	no
Interoperability	not applicable	**average**	low
Security	**high**	average	low to average

Figure 8: Comparison of key features of HPC, Grid and Cloud [5]

The hybrid approach has a promising potential by aiming to combine the individual strengths of these three paradigms and even enhance them if possible, while reducing the weaknesses. A capacity vs capability analysis of these paradigms is well-depicted in Figure 9. The Elastic Cluster introduced by study [5] is the model that combines these strengths. It must be noted that this Elastic Cluster differs by the one defined by OpenNebula as this study [5] includes: 1) dynamic infrastructure management services (DIMS), 2) cluster level services and 3) intelligent modules to bridge gap between the previous two. Elastic Clusters has the features of re-sizing to adjust to the requirements, having an automatic system that provisions further resources needed. Through modelling and utilising Workload and Resource Management Systems (WRMS) and incorporating DIMS, the approach aims to add the features of both and this is achieved by Elastic Cluster. The potentials of Elastic Cluster lie in job predictable execution (when the jobs will start) and automatic scale-down elasticity. The study [5] includes algorithms for achieving these potentials.

To realise the potentials of Elastic Cluster, the researchers have proposed to design a new distributed information service that exhibits effectiveness, reliability and scalability. Named Key Partitioned Database (KPAD), this is the distributed information service design presented. This design offers potentials as it has these features: 1) information is disturbed across multiple locations, 2) a distributed hash table (DHT) and 3) ability to manipulate tables using SQL. This new design could be safely called as a generalisation of the DHT technique [5].

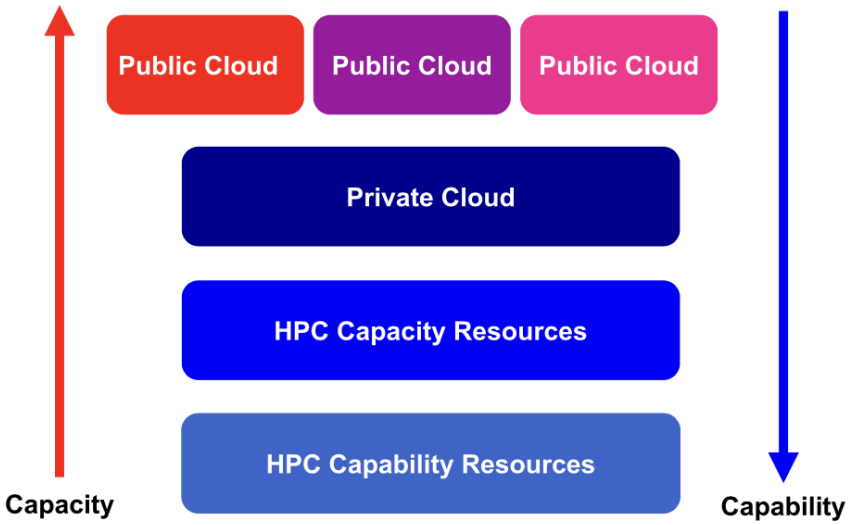


Figure 9: Resources hierarchy in hybrid architecture [5]

The DHT mechanism is implemented using an enhanced version of Content Addressable Network (CAN) [5]. Handling node departure, CAN achieves load balancing which has potential to increase speed ups but it is not without challenges as this technique does not preserve the rectangle structure of zones that a node owns. This leads to the problem of new taking over node to look after multiple rectangles. Addressing these challenges, enhancement of CAN in KPAD can be done in these ways: 1) improving availability and performance by key replication and 2) load balancing and introducing a new technique to reassign zones owned by departing nodes.This study [5] identifies three main strategies to realise the potentials of hybrid architecture proposed. These are: 1) scale up and scale down (size of Elastic Cluster) — this strategy is also known as cloud bursting, 2) scale out (spreading the load across many Elastic Clusters and 3) personal virtual cluster. This hybrid model serves as a starting point to fully tap the vision of using on-demand compute resources for heavy applications.

2.5 Fault-Tolerance approach for HPC in Cloud

The challenges and limits posed when running HPC applications in the Cloud are further complicated in situations where fault occurs. As Cloud Computing offers different solutions to high performance computing, it makes use of a high number of Virtual Machines. The presence of a large number of Virtual Machines and electronic components in HPC systems make this vulnerable to faults, as any fault would cause restarting the application and thus incurring costs in terms of time, energy and money.

In this article, we examine a study [17] that presents a proactive fault tolerance

(FT) approach for HPC systems in the cloud to cut down the wall execution time when faults occur. Surprisingly, more than 50 percent of failures on HPC systems are attributed to hardware issues (processors, circuits, memory, drive). Although, a reactive FT approach already exists but it does not work well with HPC applications. For the proactive fault tolerance approach, the researches [17] primarily focused on Message Passing Interface (MPI) applications. For performance monitoring, the control over application in dedicated HPC systems is present. However, the Cloud providers do not usually provide fault tolerance at this level [17].

The proactive FT approach by [17] basically revolves around an avoidance mechanism to lessen the negative impact of faults. Two features that assist this FT approach are system log and health monitoring facilities. Consisting of four modules, this proactive FT mechanism has 1) Node monitoring with an lm-sensor, 2) a failure predictor, 3) a proactive fault tolerance policy module and 4) controller module.

To test approach in a real cloud environment, four servers/nodes (HaaS type) were leased and all of them had the same configuration. An industrial standard virtualisation technology, Xen hypervisor [17], runs on each node where a para-virtualised guest OS is installed. Three experiments were conducted with 2, 4, 8 and 16 nodes per cluster. The application to be run is High Performance Linpack (HPL) benchmark. This HPL application was run with four different problem sizes of 2000, 4000, 6000 and 8000 on different number of nodes. From figure 10, the reduction in wall clock execution time can be noted.

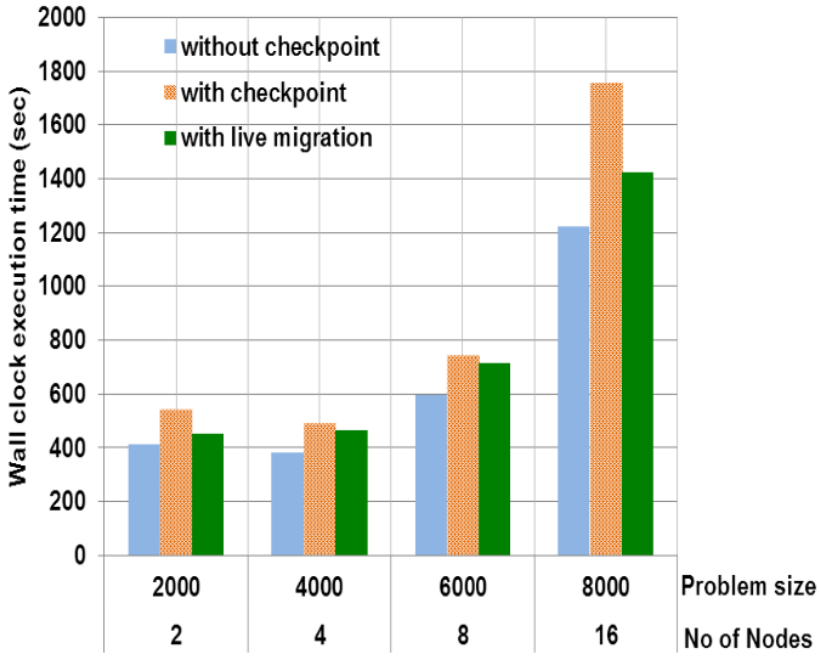


Figure 10: Performance of HPL benchmarking without checkpointing, with checkpointing and FT approach [17]

Thereby, this proactive FT approach has a high potential to reduce the execution times by having fault tolerance. The application ran is computation-intensive and thus, this mechanism opens up the potential for scientific community that require intensive computations to be done at a fast speed. Prediction techniques are employed to predict failures and this work [17] provides fault tolerance to HPC in the cloud at hardware level, also keeping the costs low. Nonetheless, this potential solution does have its limits as this is designed for users that lease HaaS (Hardware as a Service).

3 Conclusion

Cloud Computing has been gaining the attraction of HPC community as it provides on-demand, pay-as-you-go solutions as opposed to setting up the infrastructure required for an in-house HPC facility. The potential in Cloud for High Performance Computing is examined in various studies during this literature review but there are many challenges and limits present. A critical analysis of the relationship between HPC and Cloud in terms of cost, performance, environmental impact, energy usage and fault tolerance provides a comprehensive understanding in this literature study. Examining various studies shows that the advancements in Cloud are making them more suitable for running HPC

applications. Nature of HPC applications also plays a role in the performance of Cloud, as there is a difference in characteristics of applications. While some are suitable for Cloud, others are not. We aim to assist future research by presenting a holistic view of the relationship between HPC and Cloud in terms of many aspects that hold importance for both HPC users and Cloud providers. This study shall bridge the gaps in previous works and help academic community understand this complicated relationship between HPC and Cloud, and aid in developing Clouds which exhibit high performance (in all aspects) for HPC computing intensive applications.

References

References used in this literature study

[1] He, Qiming, et al. "Case study for running HPC applications in public clouds." *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing.*, 2010.

[2] Roloff, Eduardo, et al. "HPC application performance and cost efficiency in the cloud." 2017 *25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP).*, IEEE, 2017.

[3] Dillon, Tharam, Chen Wu, and Elizabeth Chang. "Cloud computing: issues and challenges." 2010 *24th IEEE international conference on advanced information networking and applications.*, IEEE, 2010

[4] Foster, Ian, et al. "Cloud computing and grid computing 360-degree compared." 2008 *grid computing environments workshop*, IEEE 2008

[5] Mateescu, Gabriel, Wolfgang Gentzsch, and Calvin J. Ribbens. "Hybrid computing—where HPC meets grid and cloud computing." *Future Generation Computer Systems*, 27.5 (2011): 440-453..

[6] Gupta, Abhishek, et al. "Evaluating and improving the performance and scheduling of HPC applications in cloud." *IEEE Transactions on Cloud Computing 4.3*, (2014): 307-321

[7] Hassani, Rashid, Md Aiatullah, and Peter Luksch. "Improving HPC application performance in public cloud." *ERI Procedia 10*, (2014): 169-176.

[8] Expósito, Roberto R., et al. "Performance analysis of HPC applications in the cloud." *Future Generation Computer Systems 29.1*, (2013): 218-229.

[9] Carlyle, Adam G., Stephen L. Harrell, and Preston M. Smith. "Cost-effective HPC: The community or the cloud?." 2010 *IEEE Second International Conference on Cloud Computing Technology and Science*, IEEE, 2010.

[10] Petri, Ioan, et al. "A HPC based cloud model for real-time energy optimisation." *Enterprise Information Systems 10.1*, (2016): 108-128..

[11] Emeras, Joseph, et al. "Amazon elastic compute cloud (EC2) versus in-house HPC platform: A cost analysis." *IEEE Transactions on Cloud Computing 7.2*, (2016): 456-468.

[12] Mariani, Giovanni, et al. "Predicting cloud performance for hpc applications: A user-oriented approach." 2017 *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, 2017.

[13] Gupta, Abhishek, et al. "The who, what, why, and how of high performance computing in the cloud." 2013 *IEEE 5th international conference on cloud computing technology and science. Vol. 1*, IEEE, 2013..

[14] Aljamal, Rawan, Ali El-Mousa, and Fahed Jubair. "A comparative review of high-performance computing major cloud service providers." 2018 *9th International Conference on Information and Communication Systems (ICICS)*, IEEE, 2013.

[15] Garg, Saurabh Kumar, et al. "Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers." *Journal of Parallel and Distributed Computing 71.6*, (2011): 732-749.

[16] Netto, Marco AS, et al. "HPC cloud for scientific and business applications: taxonomy, vision, and research challenges." *ACM Computing Surveys (CSUR) 51.1*, (2018): 1-29.

[17] Egwutuoha, Ifeanyi P., et al. "A proactive fault tolerance approach to High Performance Computing (HPC) in the cloud." 2012 *Second International Conference on Cloud and Green Computing*, IEEE, 2012.

[18] Marathe, Aniruddha, et al. "A comparative study of high-performance computing on the cloud." *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, 2013

[19] Mauch, Viktor, Marcel Kunze, and Marius Hillenbrand. "High performance cloud computing." *Future Generation Computer Systems 29.6*, (2013): 1408-1416.

[20] Lynn, Theo, et al. "Understanding the determinants of cloud computing adoption for high performance computing." *51st Hawaii International Conference on System Sciences (HICSS-51)*, University of Hawai'i at Manoa, 2018.

The Territorial Reach of the GDPR on Cloud Services

Alex Antonides
14113961
University of Amsterdam
Amsterdam, The Netherlands
alex.antonides@student.uva.nl

Marianne Hernholm
14071770
University of Amsterdam
Amsterdam, The Netherlands
marianne.hernholm@student.uva.nl

Mulham Jayab
12022985
University of Amsterdam
Amsterdam, The Netherlands
mulham.jayab@student.uva.nl

Abstract

The General Data Protection Regulation (GDPR) has a pivotal role in ensuring the right of data protection to citizens of the European Union. The GDPR claimed to have innovated its predecessor, the Data Protection Directive, by expanding on its territorial scope.
To aid the research into this topic, the following problem statement was formulated: Explore the reach of the GDPR with respect to Cloud-Service Providers.
This study has clarified the role of Cloud-Service Providers in cloud computing by identifying and describing the three main types of cloud architecture.
Furthermore, this research has clarified the role of the GDPR; describing the impact of the GDPR on Cloud-Service Providers; and which of the sections of the GDPR affects Cloud-Service Providers, with respect to the different cloud architectures.
This study has described how the GDPR affects cloud services by describing the clear distinction between the two operators and by describing their respective responsibilities, reliableness, and accountability.
The territorial reach was examined by looking into the applicability of the GDPR on Europe-based organisations. The GDPR enforced organisations to adopt their secure measures in the way of processing, controlling, and protecting personal data. The extraterritorial applicability of the GDPR was also examined. Specifically how the European Commission deemed certain countries adequate for data transfer. Due to the strict requirements, many sites from inadequate countries are now inaccessible. There have been attempts of developing frameworks to allow data transfer between countries, but theses attempts are deemed invalid by the Court of Justice.
In general, therefore, it seems that the territorial reach of the GDPR is capable of reaching countries outside the European Union, but due to the high standards, many sites are now inaccessible to Europeans. Thus, it would do good to develop a valid framework that would allow countries to transfer data to inadequate countries with respect to the user's privacy.

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

Contents

1 Introduction **3**

2 Cloud Service Providers **4**

2.1 Cloud Computing 4

2.2 The NISTs Cloud Reference Model 4

2.3 Cloud Architectures 5

2.4 Current Status and Industry Leaders 5

3 The General Data Protection Regulation and Cloud Service Providers **6**

3.1 What is the GDPR? 6

3.2 Impact of GDPR on CSPs 6

3.2.1 Sections of the GDPR affecting CSPs 6

3.2.2 GDPR considerations for different cloud architectures 7

3.3 Discussion 9

4 Data Processors and Data Controllers **10**

4.1 Distinction 10

4.2 Complications 10

4.3 Responsibilities and Obligations 10

4.4 Discussion 10

5 Europe-based Applicability **11**

5.1 Challenges and Necessary Changes 11

5.2 Data Portability 11

5.3 Data Retention 11

5.3.1 Data Breach 11

5.3.2 Discussion 12

6 Extraterritorial Applicability **13**

6.1 Cross-border Data Transfer 13

6.1.1 Privacy Shield 14

6.2 Complications 14

6.2.1 Shared Responsibility model 14

6.3 Discussion 14

7 Summary **15**

1 Introduction

The General Data Protection Regulation (GDPR) has a pivotal role in ensuring the right of data protection for citizens of the European Union (EU). The GDPR claimed to have innovated its predecessor, the Data Protection Directive (DPD), by expanding the territorial scope. This begs the question, how far is the territorial reach of the GDPR?

To aid the research into this topic, the following problem statement was formulated: Explore the reach of the GDPR with respect to Cloud-Service Provider (CSP)s.

This paper is divided into five main sections. To offer an introduction into the topic, the first section gives background knowledge into the topic of CSPs. It does this by relying on the National Institute for Standard and Technology (NIST) reference model among other sources. The succeeding section introduces the GDPR and explores sources discussing it both as a standalone regulation and with regard to CSPs. The next main section further explores the topic of how the GDPR affects CSPs. It does this by delving into the terminology of the GDPR and looking at how the GDPR distributes responsibilities and obligations for different parties. Following this, the succeeding section attempts to explore the problem statement by describing the territorial applicability of the GDPR in the European Union. Finally, the article delving into its last main section which examines the extraterritorial applicability of the GDPR.

2 Cloud Service Providers

2.1 Cloud Computing

IBM describes Cloud Computing as a structure that *"transforms IT infrastructure into an utility. It lets you 'plug into' infrastructure via internet, and use computing resources without installing and maintaining them on premises" [1].*

Russo *et al.* also defines the term in their research paper *Cloud Computing and the New EU General Data Protection Regulation* [2] as *"(..) a set of technologies and service models that allow access to a scalable and elastic pool (provisioned or released on demand) of shareable computing resources (through a common access to the service provided separately for each user)"*[2].

A Cloud-Service Provider is a company that work on making various cloud services available to customers [3]. Cloud-Service Providers make the available resources accessible to customers by claiming periodical fees for their service [1].

2.2 The NISTs Cloud Reference Model

The National Institute for Standard and Technology (NIST) created a publication defining the threshold for cloud computing architectures [4]. The publication consists of two main parts. The first regards the cloud reference model with regards to the set of roles associated with it, while the second considers it with regards to the set of activities that define its architectural components [5].

The NIST defines a conceptual Reference Model (1) which includes the five main actors in a cloud computing scheme; the cloud consumer, the cloud auditor, the cloud provider, the cloud broker and the cloud carrier. In short summation, the roles of the five actors can be defined as follows:

1. Cloud Consumer:
   (a) SaaS consumers - Use cloud services
   (b) PaaS consumers - Use cloud services for business intelligence requirements like database managing
   (c) IaaS consumers - Use cloud services for IT requirements like storage and backups
2. Cloud Auditor: Audits the cloud infrastructure of a company
3. Cloud Provider: Make cloud services available to consumers
4. Cloud Broker: Manages cloud service providers
5. Cloud Carrier: Responsible for resources that support cloud computing

[4, 5]

The architectural components of the Cloud Reference Model define the relationship between cloud providers and consumers. The model defines the following components:

1. Service deployment:
   (a) Public - Infrastructure mostly accessible for wide range of audiences like the general public
   (b) Private - Infrastructure accessible only to individual customers
   (c) Community - Infrastructure is mostly accessible to a consumer group with certain similarities (eg. security requirements)
   (d) Hybrid - Infrastructure accessible via actors like a cloud broker
2. Service Orchestration: three layers of system components needed by cloud providers for service delivery. The layers are:
   (a) Service layer
   (b) Physical resource layer
   (c) Resource abstraction and control.
3. Cloud Service Management:
   (a) Management of business requirements

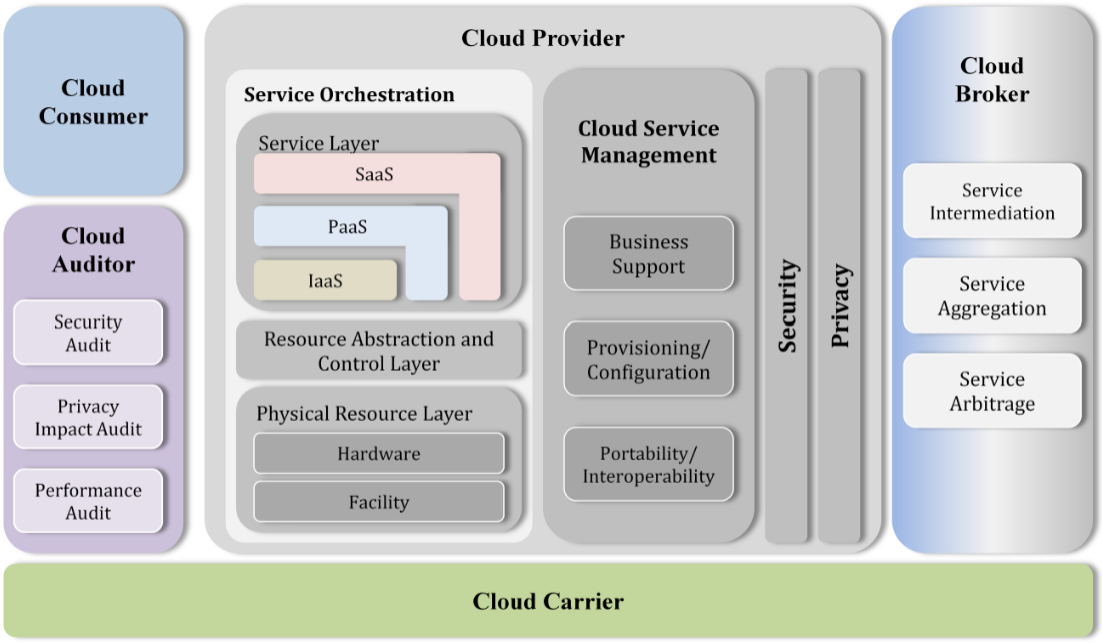


Figure 1: The NIST's Conceptual Reference Model

(b) Management of logistical requirements
(c) Management of information-related tasks and requirements

4. Cloud Security: Concerns security issues relating to cloud infrastructure
5. Cloud Privacy: Concerns the protection of the data of cloud consumers

[4, 5]

2.3 Cloud Architectures

There are three main types of cloud architectures; Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). They mainly differ in which service layer they offer services to customers [2]. IaaS schemes offers hardware and infrastructure services, PaaS schemes offers services representing the operating system layer and SaaS schemes offers services at the application layer [2].

IaaS offers resources for computing and hardware infrastructure in a virtualized environment [2]. Examples of this cloud architecture is Microsoft's Dropbox, Google's Compute Engine and Amazon's Compute Cloud. PaaS schemes offer users cloud environments for development and deployment of custom applications. Microsoft Azure is an example of a PaaS cloud architecture. SaaS offers software types (application or service) through the cloud. Example of SaaS is Salesforce [6, 2].

2.4 Current Status and Industry Leaders

In 2020 IBM claimed 25 percent of organisations had plans to move the entirety of their applications to the cloud [1]. There are numerous articles ranking the top CSPs in todays industry. Some companies who can unequivocally be identified as industry leaders are Amazon Web Services, Microsoft Azure, Google Cloud Platform and Alibaba Cloud. Computational tasks that require more computing storage and capacity than most companies can afford or find it worthwhile to obtain on-premise, present potential business opportunities for CSPs [1].

3 The General Data Protection Regulation and Cloud Service Providers

3.1 What is the GDPR?

The General Data Protection Regulation came into action in 2016 after being passed in the European Parliament, and has required organizational compliance as of March 25, 2018 [7]. It's stated intention and objective is to regulate the processing and movement of personal data by establishing rules intended to protect the people whose personal data is in question [8]. The regulation was written and passed by the European Union, but is valid for any organization whose work involve storing, collecting and targeting data from citizens withing the EU. This means that it applies to all enterprises who provide products or services, or monitor data subjects in the EU [7, 2]. Cloud providers are required to follow principles of the GDPR. The principles are *"lawfulness, fairness and transparency, purpose limitation, data minimization, accuracy, storage limitation, integrity and confidentiality and accountability"*. They are also responsible for following the regulation's requirements with regards to processing the data of data subjects [6]. Article 6 of the GDPR [8] states the terms that qualify data processing to be lawful as:

1. Processing shall be lawful only if and to the extent that at least one of the following applies:
   (a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;
   (b) processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract;
   (c) processing is necessary for compliance with a legal obligation to which the controller is subject;
   (d) processing is necessary in order to protect the vital interests of the data subject or of another natural person;
   (e) processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority vested in the controller;
   (f) processing is necessary for the purposes of the legitimate interests pursued by the controller or by a third party, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject which require protection of personal data, in particular where the data subject is a child.

[8]

The GDPR describes itself as the *"toughest privacy and security law in the world"*, and consists of 11 chapters, 99 articles and 173 recitals [8].

3.2 Impact of GDPR on CSPs

Cloud-Service Providers and their sub-contractors fall under the category of organizations who are required to to be in compliance with the GDPR [6]. Many cloud computing companies are situated outside of the EU, but have customers in the entire world. This obligates them to GDPR compliance [2]. Google is a prime example of this. This subsection explores which sections of the GDPR apply to Cloud Service Providers. And what challenges the different cloud architectures bring with them, with respect to GDPR compliance. Points a) and f) of the list above are especially relevant for Cloud-Service Provider companies. CSPs are required to obtain demonstrable consent from the data subjects whos data they are collecting and/ or processing [6]. CSPs should thus collect, store and process the data of data subjects only when there are legitimate and specific purposes for doing so. The data should be erased as soon as the previously stated term is no longer applicable [6].

3.2.1 Sections of the GDPR affecting CSPs

Researchers at the University of Piraeus in Greece published the research paper *GDPR compliance: proposed technical and organizational measures for cloud provider* [6]. The paper proposes measures for CPS companies who aim to be in compliance with the GDPR.

The article points out the main sections from the GDPR a CSP needs to consider in order to be able to be in compliance with the regulation:

1. Material and territorial scope
2. Data protection principles
3. Consent
4. Children - parental consent
5. Sensitive data and lawful processing
6. Information notices
7. Subject access, rectification and portability
8. Rights to object
9. Right to erasure and right to restriction of processing
10. Processing and automated decision-taking
11. Accountability, security and breach notification

[6]

3.2.2 GDPR considerations for different cloud architectures

The type of cloud service architecture in use has an effect on the challenge a CSP faces with respect to GDPR compliance. All cloud architecture types should comply with the GDPR [6]. Georgiopoulou *et al.* identify consideration points for different cloud architectures to assist in achieving compliance [6]. The finds of their research paper *GDPR compliance: proposed technical and organizational measures for cloud provider* [6] regarding GDPR applicability for different cloud architectures are explored below.

IaaS providers mostly offer computing infrastructure and virtualized hardware, giving customers a lot of freedom in how they wish to use said infrastructure. The providers are thus not necessarily familiar with how customers are using their services, which makes it more difficult to provide individual aid for customers. In figure 2 we see an overview of the main GDPR requirements that are applicable to IaaS providers, as identified in the research conducted by Georgiopoulou *et al.* [6].

GDPR requirement	Applicability for cloud provider
Material and territorial scope	Obligatory
Data protection principles	Obligatory
Consent	Recommendation
Children – Parental Consent	Recommendation
Sensitive data and lawful processing	Recommendation
Information notices	Recommendation
Subject access, rectification and portability	Provide information regarding the data transfers and information regarding rectification and reassure
Right to object	Recommendation
Right to erasure and to restriction of processing	Recommendation
Profiling and automated decision-taking	Recommendation
Accountability, security and breach notification	Recommendation

Figure 2: GDPR requirements for IaaS

PaaS gives customers control of the main application, but little control over the underlying environment. Figure 3 shows the main GDPR requirements that are applicable to PaaS providers, providers, as identified in the research conducted by Georgiopoulou *et al.* [6].

SaaS actors offer software application services which are often intended to process user data. SaaS cloud architectures have obligations regarding data processing-activities according to the GDPR. Figure 4 illustrates the main GDPR requirements that apply to SaaS providers, as identified in the research conducted by Georgiopoulou *et al.* [6].

GDPR requirement	Applicability for cloud provider
Material and territorial scope	Analytic information regarding the geographical information should be available online, by mail alert and official documentation
Data protection principles	Measures identify the cause of the personal data breach, mitigate adverse effects and prevent a recurrence
Consent	Recommendation
Children – Parental Consent	Recommendation
Sensitive data and lawful processing	Must employ tools for sensitive data identification and relevant measures for classifying and protecting them
Information notices	Maintain full documentation of platforms and security mechanisms
Subject access, rectification and portability	All information hosted must be exportable and readable
Right to object	Obligatory
Right to erasure and to restriction of processing	Provide tools for restriction of data storage, retention or deletion
Profiling and automated decision-taking	Recommendation
Accountability, security and breach notification	Include firewalls, network protection tools and incident management

Figure 3: GDPR requirements for PaaS

GDPR requirement	Applicability for cloud provider
Material and territorial scope	Maintain documentation of data location, information flow and alerts
Data protection principles	Measures on physical network and software level
Consent	Recommendation
Children – Parental consent	Recommendation
Sensitive data and lawful processing	Sensitive data and relevant documentation of where it could be stored from software partition level, up to physical infrastructure required
Information notices	Maintain full documentation of platforms and security mechanisms
Subject access, rectification and portability	Provide information regarding the data transfers and information regarding rectification and reassure
Right to object	Recommendation
Right to erasure and to restriction of processing	Recommendation
Profiling and automated decision-taking	Data minimization principle
Accountability, security and breach notification	Raised from SaaS providers to let know the data controller about the leakage

Figure 4: GDPR requirements for SaaS

3.3 Discussion

The General Data Protection Regulation is an extensive regulation, but its intention is ultimately to strengthen the rights data subjects have to their own data. The requirement for GDPR compliance can pose several challenges for Cloud-Service Providers, as we have briefly discussed. However, we also believe it poses opportunities for CSPs, especially with regards to trust. Some of the challenges of cloud computing are trust, security and privacy [9]. When data is stored in the premise of the CSP, users may feel a lack of control over how their data is being handled [9]. Knowing there is a regulation with strict principles that creates obligations for how Cloud-Service Providers are required to collect, store and process their personal data may lead them to feel safer in their use of Cloud-Service Providers. This increase in trust will ultimately benefit CSPs as an industry.

4 Data Processors and Data Controllers

Numerous organisations were not prepared for the obligations that were introduced by the European Data Protection Law; many would have to change their services to comply with the law and to ensure the user's fundamental right of data protection [10]. However, in cloud computing, when a subject is not compliant to the law, who would be held accountable, the Cloud-Service Customer (CSC) or the Cloud-Service Provider?

4.1 Distinction

The General Data Protection Regulation clarifies the responsibility, reliability, and accountability of the subject by making a distinction between two operators: a) `data controllers` and b) `data processors` [11]. The `data controller` determines the purpose for which and the means by which personal data is processed, whereas the `data processor` acts on behalf of the relevant controller and under their authority. The processor should only process personal data in line with a controller's instructions, unless it is required to do otherwise by law. Nevertheless, employees of the controller are not processors, they are part of the controller, unlike third parties that are contracted to process data on the controller's behalf. It would be logical to conclude that the CSC is classified as the `data controller`, while the CSP is classified as the `data processor`.

4.2 Complications

In reality, it is hard to determine whether the CSP is the data processor or the data controller, because an individual that uses third party cloud service is unlikely to be able to specify how the personal data under their control is processed [10]. This data is stored among various servers and other storage devices across the cloud. Information and personal data is is disassembled, transported from one data center to the next, reassembled, and supplied to the client on request; the client has no control how the data is processed. Moreover, some providers offer the service to store account information for account registration, administration, service access, or contact information [12]. Furthermore, there are also many consumer-oriented cloud services where CSCs are given free service, because the CSP is able to use the obtained data to help pay for them, by means of target advertising. Regardless of the operator, eventually both parties share the responsibility of the subject's data.

4.3 Responsibilities and Obligations

Under the European Data Protection Law, processors and controllers share responsibility of how personal data is processed and how the service and infrastructure are deployed and utilised; all conducted data processing is subject to the GDPR, even if this activity is not conducted within EU territory or related to EU subjects [2, 10]. That is to say, that all European-based data controllers are responsible for the data processing that is conducted inside and outside of the European Union (EU), thus they also have to be aware of their extraterritorial data processors [13, 14]. Thus, data may only be transferred to third countries, if the EU Legal Standards apply to their processing. In addition, CSPs that provide social networking services over the internet, are also responsible for managing the customers and users data and they have to comply with the GDPR regulations, since they are providing global services for users.

4.4 Discussion

As mentioned in the previous section, there are various types of cloud service architectures, thus the responsibility of controlling and processing could differ [15]. The Infrastructure as a Service architecture provides computing resources and infrastructures in a virtualised environment, therefore, the IaaS are likely to be data processors, because they are responsible for processing and handling the subject's personal data that are stored in virtual machines. However, IaaS providers are not responsible of how their system and infrastructure is used, e.g., if an enterprise does not comply with the GDPR.
We think this is a logical approach, because we agree that the provider shouldn't be responsible for the customer's content, because CSP only provide a platform to which the CSC can host their application on.

5 Europe-based Applicability

Freitas *et al.* conducted interviews with various small and medium-sized enterprises (SMEs), prior to the application of the GDPR, to examine their compliance of the approaching regulations [16]. Their study indicated that most SMEs were neither aware of the General Data Protection Regulation nor of their obligations as a data controller in terms of processing and storage principles. Furthermore, the companies that contracted third parties were unaware of the need to have their contracts compliant with the GDPR, if the third party was collecting, storing, accessing or processing personal data to supply their services. Interestingly, none of the interviewed SMEs transfers data to countries out of the European Union [16].
These SMEs would have to set up new contracts with the CSPs, in order to be compliant to the EU Data Protection Regulations.

5.1 Challenges and Necessary Changes

The GDPR affects all organisations that operate in Europe or have customers in Europe. These demands of data protection have enforced existing organisations to make changes to comply with the GDPR [17]. This has an impact on potential opportunities for the emergence of new organisations. Cloud-Service Providers and data centres have to adopt their security measures and the way of processing personal data according to the GDPR rules. Not complying with the GDPR rules may cost them very large fines of the European Union, therefore, re-engineering of existing systems in some cases is needed to ensure that they meet the GDPR requirements [17, 18]. The CSPs are obligated to provide information about data processing subjects in accordance with the corresponded GDPR articles.

5.2 Data Portability

Data portability is about the ability to move and transfer data among different Cloud-Service Providers. The CSPs have to provide the right of data portability and the ability for their customers (controllers) to retrieve and transfer the data in structured format and common technology [19, 15]. This could be done by CSPs by providing the technical capability for their customers to retrieve and transfer the data. It could be that an organisation want to transfer and upgrade a specific system or application to another cloud solution. And in order to do this the customer data held by that system could be crucial for this operation. GDPR Regulations enforce CSPs to provide the the technical capability to transfer the data. So, CSPs who are providing or wish to provide a service in the EU are obligated to guarantee this right of data portability [15].

5.3 Data Retention

Data retention is about implementing and defining some retention periods on how long the data is allowed to be stored. Under the GDPR, the personal data should not be stored longer the predefined retention periods [20, 19]. The data controllers should be able to delete this personal data completely in all stored locations. The challenge is that the data could be stored in multiple locations by CSPs. Therefore, data processors have to provide the data controllers with the technical capability to manage and control their personal data.

5.3.1 Data Breach

Under the GDPR, the data processors CSPs are obligated to send notifications for the data controllers their customers in any case of data breach without any delay. The data processors have to provide support to their customers to manage that data breach [19].

5.3.2 Discussion

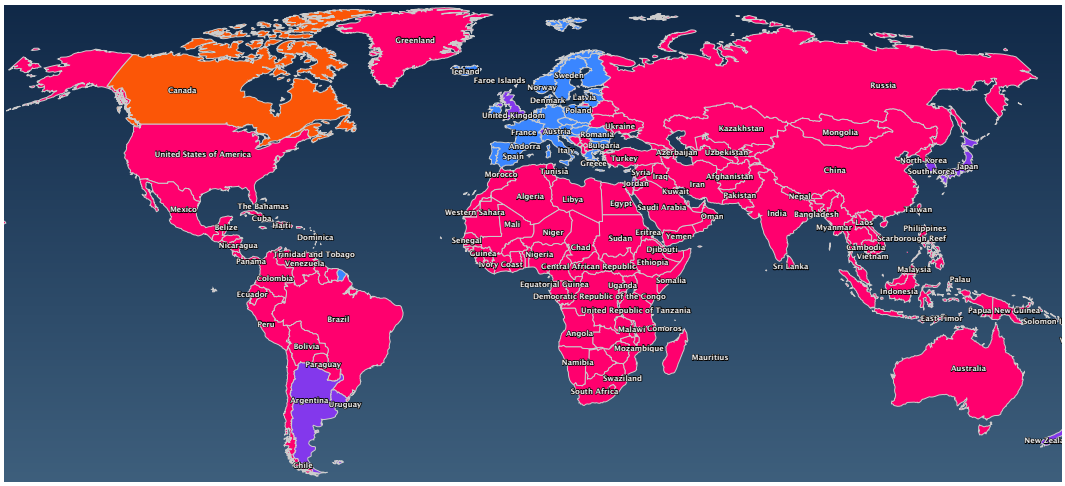
The GDPR has enforced organisations to adopt their security measures and the way of processing, controlling, and protecting personal data. We believe that some of these obligated changes that GDPR require such as data portability and data retention, could have an effective role in the development of systems used in organisations. Giving Cloud-Service Customers more flexibility to delete, transfer and move data, could facilitate and encourage organisations to use the latest technologies and update their systems. The most challenge thing is that data processors have to provide data controllers with the technical capability to manage and control their systems in more secure and flexible way. But on the other hand, these strict GDPR obligations may pose obstacles for small companies which could also be very challenging for them.

6 Extraterritorial Applicability

The General Data Protection Regulation innovated data protection regulations by expanding on the territorial scope of data protection, resulting in a wider application of the law, indicating that the GDPR is applicable outside of the borders of the European Union [21]. The territorial scope includes the location of where data may be processed. Which means that any processor that is established outside the European Union, is also subject to EU Data Protection Law, because they are active in the EU market, offering their products or services to EU citizens, thus widening the extraterritorial application of the Regulation [10]. Moreover, the territorial scope also includes the storage location where personal data may be stored of EU residents.

6.1 Cross-border Data Transfer

The GDPR distinguishes between countries outside the EU that are considered to ensure an adequate level of protection for personal data. Subjects that transfer personal data to a subject outside of the European Union, have to ensure that their data is sent to a country that is deemed adequate by the European Commission, so it does not require prior approval from the authority. The European Commission determines whether the country in question offers an adequate level of data protection for personal data flow. At the time of writing, the European Commission has so far recognised Andorra, Argentina, Canada (commercial organisations), Faroe Islands, Guernsey, Israel, Isle of Man, Japan, Jersey, New Zealand, Republic of Korea, Switzerland, the UK under the GDPR and the LED, and Uruguay as providing adequate protection [22].



Source: `https://adequate.country/`

Figure 5: Map of GDPR Adequate Countries

Figure 5 illustrates the adequate countries, where a blue indicates that it is an EU/EEA country; purple indicates that the country is adequate; orange indicates that there is a complex relationship; and red is an inadequate country.
In the absence of an adequacy designation, the controller or processor can make appropriate safeguards under the GDPR to enable cross-border data transfers to a third country or an international organisation. These safeguards can be a legally binding contract between public authorities or an approved code of conduct.

6.1.1 Privacy Shield

The European Union does not list the U.S. as an adequate country that meets the requirements of the GDPR. The U.S. Department of Commerce, the European Commission, and the Swiss Administration created the Privacy Shield, which is a framework whereby participating companies are deemed as having adequate protection, and therefore facilitate the transfer of information of registered organisations and Adequate countries [23]. This allowed organisations to have a workaround, however, on July 16, 2020, the Court of Justice of the European Union issued a judgement declaring the EU-U.S. Privacy Shield invalid.

6.2 Complications

The General Data Protection Regulations reach starts to get complicated whenever an extraterritorial Cloud-Service Customer, that isn't present in the EU, uses an extraterritorial Cloud-Service Provider, that is present in the EU. The CSP is subject to the GDPR, thus, having to change their services, affecting the CSC, even though the CSC is not subject to the European law. Popular Cloud-Service Providers like and Google Cloud claim that all of their services can be used in compliance with the GDPR [12, 24, 25].

6.2.1 Shared Responsibility model

Even though both provider and processor are held accountable for GDPR compliance. Numerous Cloud-Service Providers hold a shared responsibility model, where the CSP promises to be compliant with the security aspect of the GDPR, but where the CSC has to agree that they are responsible for how the system is used. Thus, as mentioned earlier in Section 4, IaaS are not responsible of how their system and infrastructure is used.

6.3 Discussion

The General Data Protection Regulation ensures the protection of personal data by enforcing organisations to secure their data and by blocking data transfer to inadequate countries. In our opinion, this is a good initiative, because personal data shouldn't be sent to countries that are outside of the reach of the GDPR. However, due to these strict standards, many organisations (mainly US-based newspapers) that do not wish to comply with the GDPR block people living in the EU from accessing their websites [26]. There have been attempts to set up a framework that allows data transfer to an inadequate country, but that wasn't good enough for the high standards of the European Commission. Hopefully, there will be an improved version of the Privacy Shield that allows companies in inadequate countries to still reach the EU, without putting subject's data at risk.

7 Summary

The main goal of this paper was to examine the territorial reach of the General Data Protection Regulation.

This study has clarified the role of the Cloud-Service Providers in cloud computing by identifying three main types of cloud architecture: a) Infrastructure as a Service, b) Platform as a Service, and c) Software as a Service. Whereas IaaS offers resources for computing and hardware infrastructure in a virtualised environment, while PaaS offers cloud environments for development and deployment of custom applications, and SaaS offers software types through the cloud.

The research has also clarified the role of the General Data Protection Regulation, describing the impact of the GDPR on Cloud-Service Providers and which of the sections of the GDPR affects CSPs, with respect to the different cloud architectures. We think that the GDPR is an extensive regulation with the intention to ultimately strengthen the rights of data subjects to have their own data, to which we believe that it poses opportunities for CSPs to strengthen the trust in their services.

This study has described how the GDPR affects cloud services by describing the clear distinction between the two operators: data processors, data controllers, by describing their obligations, including their responsibility, reliability, and accountability. However, in cloud services, the CSP can act as both of these operators, adding complications to the model.

The territorial reach was examined by looking into the applicability of the GDPR on Europe-based organisations. The GDPR enforced organisations to adopt their secure measures in the way of processing, controlling, and protecting personal data. We believe that such changes could have an effective role in the development of systems used in organisations. However, the strict changes may pose obstacles for smaller companies.

The extraterritorial applicability of the GDPR was also examined. Specifically how the European Commission deems certain countries adequate for data transfer. Unfortunately, due to these changes, many sites that come from these inadequate countries are inaccessible and we think that time needs to be invested in frameworks that could allow data transfer between countries, without putting the user's data at risk.

In general, therefore, it seems that the territorial reach of the General Data Protection Regulation is capable of reaching countries outside the European Union, but due to the high standards, many sites are now inaccessible to Europeans. Thus, it would do good to develop a valid framework that would allow countries to transfer data to inadequate countries with respect to the user's privacy.

References

[1] S. Vennam, "Cloud computing," Aug 2020. [Online]. Available: https://www.ibm.com/cloud/learn/cloud-computing

[2] B. Russo, L. Valle, G. Bonzagni, D. Locatello, M. Pancaldi, and D. Tosi, "Cloud computing and the new eu general data protection regulation," *IEEE Cloud Computing*, vol. 5, no. 6, pp. 58–68, 2018.

[3] J. Mathenge, "Cloud service providers (csps) explained," Aug 2021. [Online]. Available: https://www.ibm.com/cloud/learn/cloud-computing

[4] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, D. Leaf *et al.*, "Nist cloud computing reference architecture," *NIST special publication*, vol. 500, no. 2011, pp. 1–28, 2011.

[5] "What is the nist cloud computing reference," Sep 2021. [Online]. Available: https://blog.rsisecurity.com/what-is-the-nist-cloud-computing-reference-architecture/

[6] Z. Georgiopoulou, E.-L. Makri, and C. Lambrinoudakis, "Gdpr compliance: proposed technical and organizational measures for cloud provider," *Information & Computer Security*, 2020.

[7] B. Wolford, "What is gdpr, the eu's new data protection law?" 2018. [Online]. Available: https://gdpr.eu/what-is-gdpr/

[8] C. of the European Union, "General data protection regulation," 2018. [Online]. Available: https://gdpr.eu/tag/gdpr/

[9] M. Alhanahnah, P. Bertok, and Z. Tari, "Trusting cloud service providers: trust phases and a taxonomy of trust factors," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 44–54, 2017.

[10] M. Vidović, "Eu data protection reform: Challenges for cloud computing," *Croatian Yearbook of European Law and Policy*, vol. 12, pp. 171–206, 12 2016.

[11] European Commission, "What is a data controller or a data processor?" Dec 2019. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/controller-processor/what-data-controller-or-data-processor_en

[12] T. Anderson, C. Gambardella, G. Russo, M. Taggart, and L. Iannario, "Navigating gdpr compliance on aws," Tech. Rep., 4 2022. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/navigating-gdpr-compliance/navigating-gdpr-compliance.pdf

[13] T. Minssen, C. Seitz, M. Aboy, and M. C. Compagnucci, "The eu-us privacy shield regime for cross-border transfers of personal data under the gdpr: What are the legal challenges and how might these affect cloud-based technologies, big data, and ai in the medical sector?" *EPLR*, vol. 4, p. 34, 2020.

[14] M. Barati, O. Rana, G. Theodorakopoulos, and P. Burnap, "Privacy-aware cloud ecosystems and gdpr compliance," in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2019, pp. 117–124.

[15] Y. Wang and A. Shah, "Supporting data portability in the cloud under the gdpr," 2018.

[16] M. Freitas and M. Silva, "Gdpr compliance in smes: There is much to be done," *Journal of Information Systems Engineering Management*, vol. 3, 11 2018. [Online]. Available: https://www.jisem-journal.com/article/gdpr-compliance-in-smes-there-is-much-to-be-done-3941

[17] M. Kutyłowski, A. Lauks-Dutka, and M. Yung, "Gdpr–challenges for reconciling legal rules with technical reality," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 736–755.

[18] H. Li, L. Yu, and W. He, "The impact of gdpr on global technology development," pp. 1–6, 2019.

[19] A. Tolsma, "Gdpr and the impact on cloud computing," Apr 2021. [Online]. Available: https://www2.deloitte.com/nl/nl/pages/risk/articles/cyber-security-privacy-gdpr-update-the-impact-on-cloud-computing.html

[20] "Cloud computing and gdpr: What you need to know: Combell," Apr 2020. [Online]. Available: https://www.combell.com/en/blog/cloud-computing-and-gdpr/

[21] M. Goddard, "The eu general data protection regulation (gdpr): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017. [Online]. Available: https://doi.org/10.2501/IJMR-2017-050

[22] European Commission, "Adequacy decisions." [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/international-dimension-data-protection/adequacy-decisions_en

[23] European Commission, Swiss Administration, and U.S. Department of Commerce, "Privacy shield overview." [Online]. Available: https://www.privacyshield.gov/program-overview

[24] Google LLC, "Trusting your data with google cloud platform," Tech. Rep., 9 2019. [Online]. Available: https://cloud.google.com/files/gcp-trust-whitepaper.pdf

[25] Google LLC , "Safeguards for international data transfers with google cloud," Tech. Rep., 9 2021. [Online]. Available: https://services.google.com/fh/files/misc/safeguards_for_international_data_transfers_with_google_cloud.pdf

[26] J. O'Connor, "Websites not available in the european union after gdpr," 5 2019. [Online]. Available: https://data.verifiedjoseph.com/dataset/websites-not-available-eu-gdpr

The Impact of GDPR on Cloud-Based Healthcare Systems

Nina Groot
12062693
Faculty of Science
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
n.groot@student.uva.nl

Hugo Yuzhong Liu
13558366
Faculty of Science
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
hugo.liu@student.uva.nl

Dewi Spooren
12149721
Faculty of Science
University of Amsterdam
Science Park 904, 1098 XH Amsterdam
d.spooren@student.uva.nl

Abstract

After the use of internet has increased massively, the switch to cloud-based healthcare systems is a logical next step. In order to maintain the privacy and security of patients' health data, it is important to establish guidelines. From the Data Protection Directive, which was founded in 1995, the General Data Protection Regulation (GDPR) followed in 2018, consisting of a set of rules for the processing and protection of personal data. This paper reports the new changes in the GDPR, the impact of it on cloud-based healthcare systems, what aspects of the GDPR are most relevant to these systems and how healthcare organizations can comply with the most relevant regulations. The introduction of the GDPR in 2018 has changed some things within healthcare organizations. The main aspects that are impacted with regards to healthcare organizations are security, consent, privacy and the overall rights of the data subject. Because of this, data subjects are often more willing to share their personal health data because they gained trust in healthcare systems.

1 Introduction

Data grows bigger, becomes more important every day and almost every company uses data in some way. One of the most promising inventions around data has been the cloud-based storage. The switch of keeping your files on a local storage drive to a cloud-based storage has made a major difference. A device with access to the web, has access to the data, thus one of the most fundamental aspects of a cloud-based system is to ensure privacy and security (1). Since the General Data Protection Regulation (GDPR) was enforced on 25 may 2018, in order to stay compliant, some new requirements were necessary. It replaces the 1995 Data Protection Directive, which was created when the internet was still in its infancy.

All companies and organisations that deal with data relating to EU citizens must comply with the GDPR. An example of an organization where it is very important to ensure that the GDPR is followed are healthcare organizations. Medical records must be protected and it is important for patients to know that their conversations with health workers or doctors are private and will not be leaked.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Consider a scenario where these guidelines would not exist, or where these guidelines are just an advice. When insurance companies can look into all of your healthcare data, they could adjust their prices, based on someones health history. Higher the price for someone who is known to have health problems. Or when conducting a job interview, the recruiter could pick people based on their health conditions. This might benefit the insurance company or the recruiter, but it is a serious form of discrimination. Because of these reasons and many more it is extremely important to protect these private data.

In this literature study the impact of the GDPR on cloud-based healthcare systems will be evaluated. First, an introduction to cloud-based healthcare systems and to the GDPR will be given, along with the rise of data protection, where-after the main changes of the GDPR in contrast to the Data Protection Directive will be reported. Afterwards the effects that arose after enforcing the GDPR on cloud-based healthcare systems and some sub-sectors will be presented. In the next section, the data protection impact assessment of the GDPR will be discussed. In the end, the impact of the GDPR on cloud-based healthcare systems will be concluded.

2 Introduction to cloud-based healthcare systems

A cloud-based environment for healthcare systems allows the sharing of someones health records among different domains, which can be very useful. There is a large amount of health records which all need to be stored somewhere. Cloud-based systems have a lot of advantages. Using a cloud-based system has proved to be scalable and thus can mage large amounts of data (2). Since healthcare data is extremely large, a scalable storage system is very important. Furthermore it allows sharing of data in a very easy and timely manner. There are three different types of cloud models which are employed in healthcare systems, the private, public and hybrid cloud (3).

A private cloud is the most secure cloud out of the three. You cannot enter via the internet and it can only be accessed by recognised personnel of the healthcare organization. It is often used for Electronic Medical Records (EMRs) of patients. These kind of clouds are provided by a cloud provider such as Amazon Web Services (AWS), however only accessible for one specific organization and where the infrastructure can be on-premise or at the cloud provider (4) .

The second type of cloud is the public cloud. As can be concluded from the name, this cloud is public to other organizations. Using a public cloud, healthcare organizations can make use of a shared online infrastructure where for example, they can share EMRs among different healthcare organizations. Such an infrastructure is often also provided by a cloud provider such as AWS (4). The advantage of a public cloud is that you only pay for a used capacity per month, so you can save money. Furthermore a public cloud is highly scalable.

A hybrid cloud is a combination of both the private and public cloud. It combines the benefits of both the private and the public cloud (3). It make sure that the organization can store private data in their own network, but still get the benefits of a shared cloud. In figure 1 an example of a hybrid cloud can be seen.

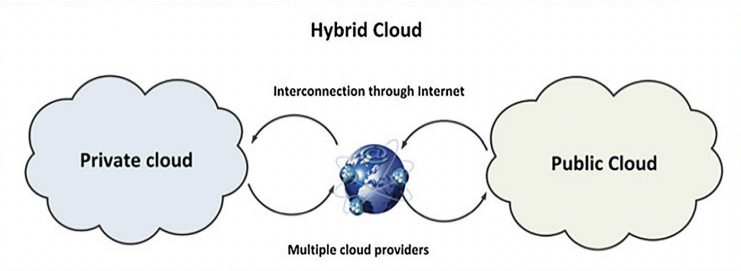


Figure 1: Example of hybrid cloud. From Security and privacy issues in e-health cloud-based system: A comprehensive content analysis, by Azeez, N. A., & Van der Vyver, C. (2019). Egyptian Informatics Journal, 20(2), 97-108.

The use of cloud-based systems in healthcare has a lot of advantages, however there are security and privacy issues that need to be addressed. Since the adoption of the GDPR, certain things need to be changed and guidelines need to be followed.

3 Introduction to data protection and the GDPR

3.1 The rise of data protection (in healthcare)

As was mentioned in the introduction, the GDPR replaced the Data Protection Directive which was enforced in 1995. However, the idea that data must be protected, arose much earlier than 1995. In 1968, at the United Nations International Conference on Human Rights occurred the first international discussion about the protection of data (5). In 1980 the OECD, which is an organisation for Economic Co-operation and Development, created the Guidelines on the Protection of Privacy and Transborder Flows of Personal Data (Guidelines). That was a step in the right direction, however these guidelines did not yet had legal force, so it was only an advice.

After the Guidelines, there became the Directive (Data Protection Directive) which resulted from the elimination of the distinction between public data protection and private data protection. The Directive applied to all sixteen EU members and was adopted around 1995. According to the Directive, personal data may be used only for the legitimate purposes for which it was collected, and kept in a form that does not permit identification of individuals longer than is necessary for that purpose (5). Furthermore according to Cate (5): "the Directive excepts only the "processing of personal data" that is performed by a "natural person in the course of a purely personal or household activity".

When thinking of data protection, one important aspect is ethics. According to the Oxfords dictionary, ethics is "the moral principles that govern a person's behaviour or the conducting of an activity". One of the earliest expressions of ethics in the medical world is the Hippocratic Oath. The oath has been written by the famous Greek Hippocrates between the fifth and third centuries BC. Hippocrates is considered to be the founder of modern western medicine. The Hippocratic Oath is an oath of ethics taken by physicians. According to Séroussi et al. (7): "The oath is build on four pillars, autonomy, e.g., patients but also physicians should keep their autonomy of thought, intention, and action when making decisions regarding health care procedures; (ii) justice, e.g., burdens and benefits of health care procedures, especially treatments, must be distributed equally to be fair with all players involved; (iii) beneficence, e.g., health care procedures are provided with the intent of doing good for the patient involved; and (iv) non-maleficence, e.g., health care procedures should not harm the patient involved."

Inspired by medical ethics, ethics in health informatics (EHI) arose, which was mainly concerning data privacy in healthcare (7). Before the internet, there was one doctor for one patient which maintained the security and privacy of their patient records, because it was unethical to share this private information. With the rise of electronic health records, which are kept in cloud-based systems, it became necessary to ensure the privacy of these records, since data can be shared among different members of a healthcare team. Because of this, the Directive which was adopted around 1995 became extremely important.

However, the internet kept evolving and thus also the guidelines on privacy. Internet in relation to healthcare is also a growing sector. Think of online doctors, at any moment you can get in contact with a nurse or a doctor, or the fact that you can order your pharmacy prescriptions online. There even exists an online AI doctor, which can give a diagnose based on your symptoms using AI-algorithms (11). Some of these systems would require storage of data in a cloud, for example the online pharmacies, whereas others might not keep track of data such as the AI doctor. But in order to require all companies and organizations to be compliant with data protection rules, the GDPR was declared in 2018.

3.2 The GDPR and its differences with the Data Protection Directive

As mentioned above there are two main set of rules relating to data protection and privacy, the Data Protection Directive (Directive) and the General Data Protection Regulation (GDPR). From the Directive, the GDPR followed. In this subsection we will dive into detail on the two different regulations and will discuss the differences between them.

The GDPR is about the protection of data with regard to the processing of personal data (5). The rules are set to secure personal data across EU countries and its citizens. In contrast to the Directive, all companies and organisations, also the ones that are not located in the EU, that deal with data relating to EU citizens must comply with the GDPR. Before diving into the regulations, it is important to understand the term 'personal data'. According to the handbook on European data protection law (8): "personal data is defined as information relating to an identified or identifiable natural person. It concerns information about a person whose identity is either manifestly clear or can be established from additional information." It covers various types of information, such as name, date of birth, address or phone number, where it is known to which person it belongs to or you can use the information to find the person.

Article 9 of the GDPR expanded the personal data definition with special categories of personal data (6). As stated in article 9, it is forbidden to process personal data which reveals racial or ethnic origin, political opinions, political opinions, religious or philosophical beliefs, or trade union membership. Furthermore it is prohibited to process genetic or bio-metric data for the purpose of identifying a person, or data concerning health or concerning a person's sex life or sexual orientation. This definition is also very important for all sorts of healthcare systems including the cloud system. Healthcare systems should not contain or process data about someones health which could be used to link it to a particular person.

The GDPR consists of 7 principles related to the processing of personal data which can be seen in figure 2. In order to fully understand the principles it is important to know and be able to distinguish three definitions which are stated in the GDPR and relate to different roles in the process of processing data. They will be described with respect to the roles of data processing in cloud-based healthcare system. Article 4 of the GDPR describes the three different roles, the controller, the processor and the data subject (6). The patient, who receives medical treatment or counseling, is the data subject. The controller determines the purposes of the processing of the data, this would be the healthcare institution. The data processor in this case is the cloud service provider (12).

Figure 2: Principles of the GDPR. From the GDPR Compliance: Proposed Guidelines for Cloud-Based Health Organizations, by Georgiou, D., & Lambrinoudakis, C, 2020. In Computer Security (pp. 156-169). Springer, Cham.

The principles will be briefly described below, with relevance to cloud-based healthcare systems. The first principle guarantees that any processing of personal data is lawful, fair and transparent in

relation to the data subject (6). In connection with cloud-based healthcare systems this means that the data subject, in this case the patient, is aware of the data being processed and why this is necessary. Furthermore, the data should be limited, data cannot be collected without an appropriate purpose, the controller, in this case the healthcare institution is responsible for this. In order to ensure data minimization the controller needs to check whether the purpose could be achieved with a smaller collection of the data. Moreover, the controller should ensure the accuracy of the collected data. It is very important that healthcare information is accurate, if a patients allergies are not carefully written down, it could result in their death. In addition to, with respect to the purpose of the data collection, the data should only be stored for as long as needed. Another principle is about the integrity and confidentiality, in relation to cloud-based healthcare systems, it is very important to limit the access to the data and the personal health data should be encrypted. At last, the controller should be accountable for the processing and needs to be able to prove the processing is lawful.

One of the most important differences between the Data Protection directive and the GDPR is that the GDPR is a regulation instead of a directive, because of this it became an enforceable law for all EU countries. Before the GDPR, the Directive was not necessarily legally binding, the EU member states must transpose the directive into a law. Nowadays, companies can be made accountable to the fact they are not following data protection rules. Another change is about to which countries the GDPR applies to. Unlike the Directive, for the GDPR it does not matter whether the personal data is located within or outside a European country; the one condition is that the data concerns an EU citizen (5). Another change is the addition of the data processor role. We briefly mentioned the different roles in the EU data protection regulation (controller, processor and data subject). Before the GDPR the data processor role did not exist. The data owner had the responsibility of the protection of data. Nowadays, it is a shared liability between the data owner and the cloud service provider.

4 The impact of the GDPR on cloud-based healthcare systems

4.1 Most relevant aspects of the GDPR and how to comply

To find the impact of the GDPR on cloud-based health care systems, first it has to be decided which aspects of the GDPR are most relevant to cloud-based health care systems. This section describes the most relevant regulations of the GDPR and how cloud-based healthcare systems/organizations can comply with these regulations.

4.1.1 Security

One of the most relevant aspects in cloud-based healthcare systems is security (9). It is important that the data is protected properly. The data that is involved in healthcare, for example health data, genetic data or bio-metric data, is very sensitive, a data breach could lead to unfortunate events. Thus the method that is used to secure this data is critical. According to the principle of integrity and confidentiality of data, the patient of a healthcare organization expects the organization to protect his or her personal data by enforcing appropriate organizational and technical protection (6). This protection concerns the overall organization and operation of the healthcare organization on data protection issues. Some examples are the existence of confidentiality clauses and its training, the existence of relevant procedures and the overall compliance of the health unit with the GDPR. This minimizes the possibilities of an unauthorized access or accidental disclosure of data, along with the unauthorized or accidental alteration of data.

Two methods of securing the processing of personal data have been mentioned in the GDPR article 4 and 6, namely pseudonymization and encryption (6). To pseudonymize data, the information that can identify a person is replaced by "pseudonyms". Encryption is a method where information is transformed into a secret code. The data is not accessible and only with a particular key it can be decrypted again.

In addition to these two methods, Butpheng et al. presented an solution for the security of cloud-based E-health systems using a five layer system (15). They identify an E-health system as “the ability to seek, find, understand and appraise health information derived from electronic sources and acquired knowledge to properly solve or treat health problems" They conducted a literature review and provided a system based on all security aspects. The system consists of the following layers:

- **Device layer**
  Designed to prevent malicious attacks on the data storage system. In this layer the devices can perform functions like detection and monitoring.
- **Communication and service layer**
  This layer allows different devices to connect with each other and is used against sniffing attacks and adulteration.
- **Network layer**
  This layer can connect to the cloud system and is used as a secure routing.
- **Cloud layer**
  Protection of stored and processed data against unauthorized users. The layer processes, stores and monitors data collected by the device layer.
- **Application layer**
  This is the layer where the user communicates with the E-health system and is used to provide the user against bribery of information.

4.1.2 Consent

Another important aspect in healthcare systems is the request for consent (10). It is crucial that the patient knows which data is being processed. The processing of special data categories (Article 9, (6)), which health, genetics and bio-metric data also belong to, can only be processed when the subject gives explicit consent. There are three main types of consent, per type it varies how to get consent of a patient:

- **Informed consent**
  According to the GDPR (6), for consent to be informed, the data subject should be aware of the identity of the controller and the purposes of the processing. Consent should not be regarded as freely given if the data subject has no genuine or free choice or is unable to refuse or withdraw consent without detriment. The consent can be in either oral or written form, for example by the subject answering a specific question about their willingness to participate.
- **Explicit consent**
  To obtain explicit consent, on the other hand, the data subject has to give an explicit affirmative action. This can, for example, be done by answering a specific question, in oral or written form, about their willingness to participate. The important difference is that the data subject actually has to perform an action, like clicking a button or ticking a box, to give the consent.
- **Broad consent**
  To give broad consent, the subject should agree to a broad set of potential secondary future uses of his or her data under a particular government framework (16). This type of consent has been widely embraced as the standard practice in genetic registries and bio-banks. It is also generally used for most big data projects where their most innovative secondary uses can't be specified by the time of data collection.

The GDPR enhances the citizen's rights when it comes to the process of consent for the collection, use and sharing of their personal data. The regulation states that consent is the main legal base to process this kind of data. There are several conditions for this consent, namely it should be explicit and unambiguous, freely given, specific, informed and signified. In healthcare, it is important that this consent also incorporates the case of many potential transfers of health data, this includes cloud storage and international data transfers. The patients need to decide if they give consent to the collection of their personal information. In healthcare, this consent could be asked for by a button or by a ticking box, along with a clear, specific and targeted information.

4.1.3 Rights of data subjects

A third important aspect is the fact that the GDPR strengthens the rights of the data subjects. Even though the rights of data subjects were already there in former legal texts of case-law, the GDPR improves them because it lists them in clear terms within other data protection rights and obligations (9). Article 15 of the GDPR requires data subjects to have access to information about themselves.

Besides that, the data subject also has the right to receive copies of it. Furthermore, article 18 enables the patients to ask their data to be restricted, meaning no processing of the data only storage the data (6). In a research regarding a cloud service, participants were worried that the controller (the healthcare organization) would access their patient data, outside of their treatment, which is against the right of the data subject (12). Article 34 prevents this by reporting such misuse to the authority and by immediately notifying the data subject.

In order to earn the trust of the data subject, and of course comply with the GDPR regulations, it is important to comply with the rights of the data subject. Because of this, Georgiou et al. (1) proposed a procedure of different steps in order for cloud-based healthcare organizations to manage requests of data subjects in a fair and transparent way. The procedure consists of the following steps:

- **Collection of data subject's request.** The data subject can send their request via mail, e-mail, via the website of the cloud-based healthcare organization or physically.
- **Identification and information of the data subject for the reception of the request.** The person responsible of the request should identify the data subject and inform them upon reception.
- **Registration of the request in the requests' record.** All requests for the cloud-based healthcare organization should be recorded in one file.
- **Forwarding the request to the data protection officer.**
- **Evaluation of the request** The data protection officer assesses whether additional information from the data subject is needed in order to proceed with the request.
- **Requesting additional information from the data subject**. If needed
- **Informing the data subject of a charge to process the request.** If a charge is needed for the request, inform the data subject about it.
- **Performing the required actions.** Satisfy the right of the data subject
- **Justified information to the data subject for delaying the satisfaction of their request.** When a delay occurred, inform the data subject
- **Informing the DPO regarding the implementation.** Once the request is completed inform the data protection officer.
- **Prepare the response document for the data subject.** the answer given to the data subject regarding the fulfillment or not of the request must be documented
- **Informing the data subject regarding the fulfillment or not of the request.** Response to the data subject via a letter, electronically or orally.

4.1.4 Privacy policies

According to the GDPR (6), the cloud provider that collects or processes personal data should take further action to protect it. This cloud provider usually acts as data processors on behalf of their users. To properly protect their users' data, the types of sensitive data that are processed should be recognized and analytically described. This should be done in the security policy of the cloud and should also provide the reasoning for their necessity. Furthermore, a short format of the privacy policy should be freely accessible to patients. This short format ought to contain the basic information and clear pointers on how to obtain the full privacy policy.

The question remains, how can cloud-based healthcare organizations ensure the privacy of a data subject. Controlling and limiting the access to personal data is very important in relation to privacy. One tool which can be used to ensure privacy is the privacy awareness cycle (15). Here, personal data is presented in the form of life cycle stages. The different stages can be seen in figure 3. Initiation represents the processing of the data, following collection where the data is gathered together, retention means the structuring and storing of the data, following access, which represents how to consult the data. Disclosure is when the data is made available to third parties and usage represents what is done with the data. In the end, destruction represents the act of erasing the personal data.

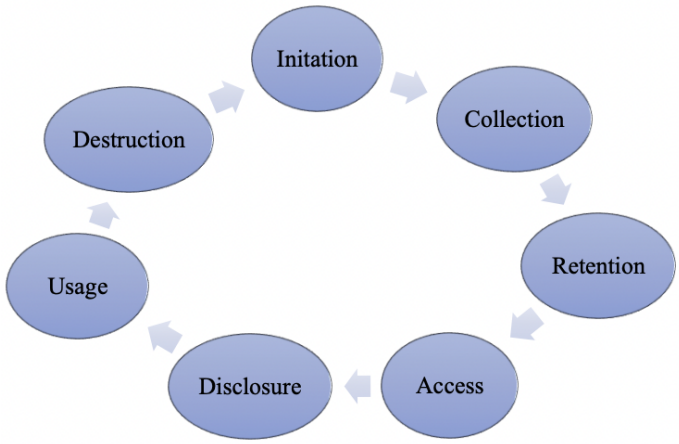


Figure 3: Privacy awareness cycle. From Butpheng, C., Yeh, K. H., & Xiong, H. (2020). Security and privacy in IoT-cloud-based e-health systems—A comprehensive review. Symmetry, 12(7), 1191.

4.2 The impact of the GDPR on certain sub-sectors of healthcare systems

In order to take a closer look at the impact of the GDPR on healthcare systems, we study two of the sub-sectors as case studies: health-related data processing for research purposes and mobile health applications.

4.2.1 Health-related data processing for research purposes

In healthcare systems, healthcare research is undoubtedly an integral sub-sector that plays a role in improving the quality of healthcare practices. Health-related data processing for research purposes is thus subjected to the impact brought by the GDPR.

Under the new regulation, researchers are facing obligations for the protection of health-related data when conducting research; such obligations can be an obstacle, in terms of time and resource allocation (17). To adapt to the new paradigm, the whole research architecture needs to adopt a series of technical and organizational measures. In addition, these measures cannot be standardized for general purposes, because each project has its own specific situations that need to be dealt with individually and can also evolve as the project proceeds. Translated into a researcher's daily routine, this piece of work could be a new combination of procedures, contacts, and administrative activities, which can be time-consuming and resource-demanding for both the researcher and the research institute.

Despite a burden the aforementioned obligations might seem to be, Amram, D. (17) sees the implementation of the GDPR as an opportunity to enhance the EU values and protect fundamental rights; after all, science serves human being, not vice versa – this is particularly relevant in the context of health-related data.

The author further proposes a model incorporating several features to achieve a standard accountability in health-related data research:

- Ethical-legal specialists should be involved in research initiatives to assist in the creation of an ethical-legally compliant ecosystem.
- To achieve ethical-legal compliance, research institutes should implement appropriate and effective technical and organizational procedures.
- Data availability, confidentiality, and integrity should be guaranteed by the IT infrastructure and data management plan.
- Data processed for healthcare and scientific research should be segregated and subjected to separate security protocols.
- Data flows should be controlled both between partners and inside research teams.

- In the information provided to data subjects, data flows should be tracked and described.

- The coordination between various legal limitations, protocols, and standards should be assessed in terms of risk and managed throughout the research's life cycle.

The article manages to highlight several practical issues that researchers may need to deal with when complying with the GDPR and other ethical requirements.

Nonetheless, there are not sufficient stakeholders (from a variety of backgrounds) involved in the proposed model. Moreover, the article does not discuss possible effectiveness and limitations that the model might have.

4.2.2 Mobile health applications under the GDPR

Mobile health (m-health) applications leverage medical devices and sensors as well as cloud technologies to deliver healthcare services, such as fitness trackers and surgical rehabs (18). However, these services come with a price – a large amount of medical data can be vulnerable if not protected properly. For instance, a patient's health data run the risk of getting exposed to unauthorized parties via the usage of cloud technologies, which, according to Forbes, are used by 83% e-healthcare service providers in some capacity (19).

As summarized by (20), there are three major threats confronting medical data: tampering with medical data (integrity issue), loss of data (availability issue), and unauthorized disclosure of data (confidentiality issue). The GDPR thus comes into play to ensure that adequate safeguards are in place to secure medical data in healthcare systems, including m-health apps.

An m-health project, WELCOME, is presented by Mustafa U. et al. in (18) as a case study to evaluate its privacy aspects in compliance with the GDPR. WELCOME provides an integrated healthcare solution based on the following devices (Table 1):

Table 1: WELCOME healthcare solution

Device	**Functionality**
A wearable vest that monitors a large number of sensors	Various physiological data, including chest sounds, heart rate and rhythm, and oxygen saturation levels, are measured by each sensor.
Medical sensors	Monitor blood trends such as glucose, blood pressure, and temperature on a regular basis
Inhaler devices	Assess the patient's medical adherence as well as their inhaling technique
Remote monitors	Track and analyze the patient's multi-parametric data, including physiological, environmental, and emotional data, to create personalized integrated care plans

On top of the devices, a system architecture is built, consisting of several core modules such as patient hub, cloud processing, and end user applications (see Figure 4).

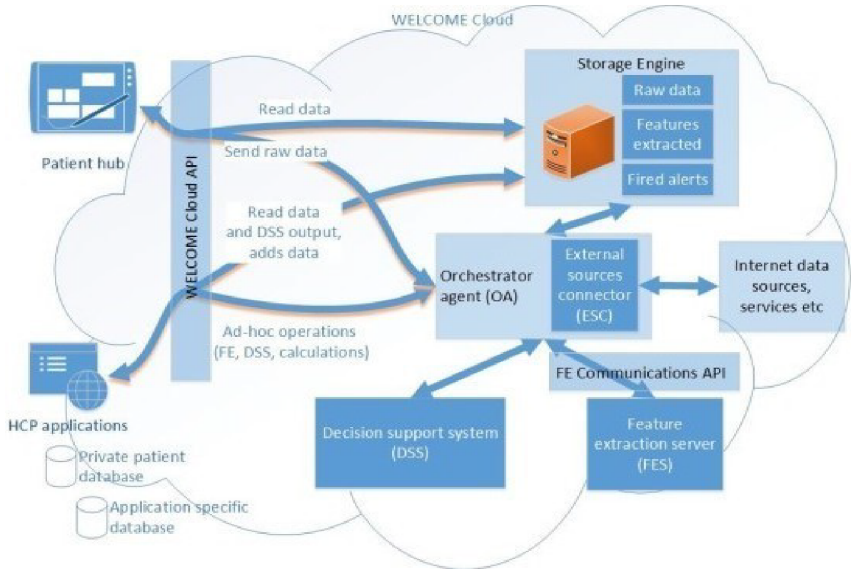


Figure 4: WELCOME core system architecture (18)

- **Patient hub**. The patient hub is a mobile platform that manages the patient's interaction as well as the data transfer. It consists of all software and hardware components that interface with the WELCOME vest and medical sensors in order to collect data, pre-process biosignals, and then send it to the WELCOME cloud.
- **Cloud processing**. Storage server, feature extraction server, decision support system, and orchestrator agent make up the system's cloud.
- **End user applications**. WELCOME applications are used to keep track of the patient's mental, behavioral, and overall health. They provide remote monitoring information and alert medical workers to potentially dangerous circumstances.

To comply with the GDPR, such an m-health project needs to follow these requirements:

- Patients must be able to examine their data and request that any faulty measurements collected by the sensors be deleted or modified.
- Patients must be notified about the sort of data acquired from the sensors they are wearing, as well as the length of time this data will be used for processing.
- Only data that is essential for the application's functionality will be collected. Only limited access is granted in order to enable the app's functionality.
- Patients must be fully informed about the security measures in place for their data in transit or in third-party storage.
- The data gathered must not be used for marketing or profiling. The patient's informed medical consent is required before any medical data is collected or stored. Patients must be made aware of the risks and benefits of utilizing a mobile health app.
- To protect themselves from device impersonation attacks, all devices must have a set of proper security features. Mobile devices that run the patient hub application can only be accessed by an authorized user.
- Authentication of the devices that share medical data is required so that only authorized devices can send and receive medical data.
- To ensure that data kept in whatever form has not been tampered with, proper security methods are required. To prevent unwanted access to data in transit as well as data in backups, proper security techniques such as encryption should be employed.

Mustafa U. et al. further describe and assess various security and privacy methods deployed in different layers of the WELCOME architecture. These mechanisms include consent management,

data minimization, data retention period, security measures, data anonymization, role-based access, and authentication and authorization process.

At last, this paper offers suggestions for developing a secure and compliant mobile health application:

- **Consent**. As discussed before, prior to using the medical application, the user must give their informed consent to process their medical data. Any prospective data transmission, including international data transfers and cloud storage, should be covered by the consent. Because the data controller now bears the burden of proof, it is critical that they demonstrate that the patient has granted consent.
- **Authentication**. Authentication is the foundation of access control and audit tracking, thus identifying m-health users is crucial. Continuous authentication is essential to ensure that the device holder is the same person who authenticated the first time. This can be accomplished through bio-metric sensing.
- **Secure application development**. While developing any medical application, secure app development procedures must be performed in accordance with the code of conduct for privacy in health applications (4). It is critical to detect and mitigate personal data protection risks. A risk assessment should be carried out, taking into account all of the services and partners with whom the application trades data.

This paper provides a clear case study to showcase what privacy and security requirements an m-health application faces under the GDPR and how it should be implemented to adapt to the new regulation.

However, the recommendations of a privacy framework proposed in the paper are not substantial, nor are they novel enough due to the considerable overlap between them and the GDPR requirements, e.g., the suggestion that medical data should not be provided to any third party for commercial use.

4.3 Data protection impact assessment required by the GDPR

As privacy concerns spread throughout society, particularly in the healthcare sector, the importance of the Data Protection Impact Assessment (DPIA) grows, as the GDPR also requires organizations to conduct a DPIA when a processing activity is likely to result in "in high-risk to the rights and freedoms of natural persons" (21).

A DPIA is a process aiming to minimize risks with respect to privacy, security, and reputation, comply with the data protection legislation, and increase trust among data subjects and stakeholders. Clearly, providing cloud-based health services involves "high-risk", given that they process a large volume of sensitive (health) data, and the DPIA is a critical step in protecting patients' privacy and rights.

Georgiou, D. et al. (22) present the main steps of a DPIA, by employing the methodology of the Privacy Impact Assessment, Commission Nationale de l'Informatique et des Libertés (PIA-CNIL) (23). The steps and the transition from one step to another are shown in Figure 5.

- **Context**. This step defines the context of the personal data processing in question. It attempts to outline the personal data processing process, including personal data categories, processing purposes, data processing methods, personal data supporting assets, data subjects, etc.
- **Controls**. This step identifies the existing and planned (procedural, technical, and organizational) controls that are required for securing data, treating privacy risks proportionately, and complying with legal obligations. This stage necessitates the rationale and explanation of decisions made about the data collection purpose, storage period, and quality.
- **Risks**. This step evaluates the privacy risks posed by data processing and ensures that they are appropriately addressed. Several issues should be addressed at this stage, including risk sources and descriptions of their capabilities; feared events; threats to personal data and assets that could result in feared events; and risk level determination.
- **Decision**. This step is called risk management decisions. Its goal is to analyze the outcomes of the previous steps, assess the risk level and existing controls, and determine whether they are acceptable.

Figure 5: General approach of the privacy impact assessment, Commission Nationale de l'Informatique et des Libertés (PIA-CNIL) methodology

Further, a case study of a cloud-based hospital information system is carried out by Georgiou, D. et al. In the case study, by identifying the purposes of processing and conducting a gap analysis, the paper presents the steps for conducting a DPIA that ensures the security and privacy aspects of the hospital information system, specifically the GDPR principles, such as purpose limitation, data minimization, accuracy, accountability, the lawfulness of processing, and the user consent (24).

It is valuable of this paper to identify the most important organizational and legal requirements that cloud-based healthcare organizations must meet under the GDPR. The study is relevant to any personal data processing since it helps to structure the process, and to bring data protection concerns to a broader audience.

5 Discussion and Conclusion

In summary, the introduction of the GDPR in 2018 has changed some things within healthcare organizations. The main aspects that are impacted with regards to healthcare organizations are security, consent, privacy and the overall rights of the data subject. We investigated multiple ways to comply with the GDPR rules, such as pseudonymization, encryption, the privacy awareness cycle and a five layer system to ensure security (15).

The main difference of the Data Protection Directive from the GDPR is that the Data Protection Directive was not necessarily legally binding. Each European member country had to create the data protection law, it did not apply to them automatically. The advantage of the GDPR is that it applies to all entities that use data from the citizens of the EU, even if the organization or company is located outside Europe. which means that organizations are forced to address their guidelines and ensure privacy and security. Because of this, data subjects are often more willing to share their personal health data because they gained trust in the healthcare organization, since they are obligated to follow the data protection rules. This is a huge advantage for the healthcare organization themselves, but also for researches in the field of health, who rely on personal healthcare data.

Through the case studies of two sub-sectors of healthcare systems, i.e., health-related data processing for research purposes and mobile health applications, we find that organizations are facing more obligations under the GDPR, and that while adapting to the new regulation, organizations can take it as an opportunity to contribute to a more secure environment for data processing in cloud-based healthcare systems.

Finally, we examine the steps of the data protection impact assessment in order to present methodologies to assess and minimize privacy risks in the cloud-based healthcare systems as well as to comply with the GDPR.

In this paper, we do not cover more sub-sectors of healthcare systems to formulate a more comprehensive perspective of the GDPR's impact, due to the scope of our topic, and to the situation that the implementation of the GDPR is still at a relatively early stage and its impact on the sub-sectors is yet to take place. Therefore, for future work, it would be interesting to investigate the impact of the GDPR on more sub-sectors of healthcare systems.

References

[1] Georgiou, D., & Lambrinoudakis, C. (2020). Compatibility of a Security Policy for a Cloud-Based Healthcare System with the EU General Data Protection Regulation (GDPR). Information, 11(12), 586.

[2] Abbas, A., Khan, M. U., Ali, M., Khan, S. U., & Yang, L. T. (2015, August). A cloud based framework for identification of influential health experts from Twitter. In 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom) (pp. 831-838). IEEE.

[3] Azeez, N. A., & Van der Vyver, C. (2019). Security and privacy issues in e-health cloud-based system: A comprehensive content analysis. Egyptian Informatics Journal, 20(2), 97-108.

[4] Cloud computing Service, Explore our solutions. Retrieved on 29 May 2022 from `https://aws.amazon.com/?nc2=h_lg`

[5] Cate, F. H. (1994). The EU data protection directive, information privacy, and the public interest. Iowa L. Rev., 80, 431.

[6] European Parliament and Council of the European Union. 2016. Directive 95/46/EC (General data protection regulation). Retrieved on May 17, 2022 from `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679`

[7] Séroussi, B., Hollis, K. F., & Soualmia, L. F. (2020). Transparency of health informatics processes as the condition of healthcare professionals' and patients' trust and adoption: the rise of ethical requirements. Yearbook of Medical Informatics, 29(01), 007-010.

[8] Council of Europe Handbook on European Data Protection Law 2018 Edition. Retrieved on May 23, 2022 from `https://www.echr.coe.int/Documents/Handbook_data_protection_ENG.pdf`

[9] Georgiou, D., & Lambrinoudakis, C. (2020). GDPR Compliance: Proposed Guidelines for Cloud-Based Health Organizations. In Computer Security (pp. 156-169). Springer, Cham.

[10] Muchagata, J., & Ferreira, A. (2018, October). Translating GDPR into the mHealth Practice. In 2018 International Carnahan Conference on Security Technology (ICCST) (pp. 1-5). IEEE.

[11] Symptomate, check your symptoms today. Retrieved on 23 May 2022 from `https://symptomate.com/nl/`

[12] Chomutare, T., Yigzaw, K. Y., Olabarriaga, S. D., Makhlysheva, A., de Oliveira, M. T., Silsand, L., ... & Bellika, J. G. (2021, March). Healthcare and data privacy requirements for e-health cloud: A qualitative analysis of clinician perspectives. In 2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM) (pp. 1-8). IEEE.

[13] Ekonomou, E., Fan, L., Buchanan, W., & Thuemmler, C. (2011, November). An integrated cloud-based healthcare infrastructure. In 2011 IEEE Third International Conference on Cloud Computing Technology and Science (pp. 532-536). IEEE.

[14] Khan, F. A., Ali, A., Abbas, H., & Haldar, N. A. H. (2014). A cloud-based healthcare framework for security and patients' data privacy using wireless body area networks. Procedia Computer Science, 34, 511-517.

[15] Butpheng, C., Yeh, K. H., & Xiong, H. (2020). Security and privacy in IoT-cloud-based e-health systems—A comprehensive review. Symmetry, 12(7), 1191.

[16] Politou, E., Alepis, E., & Patsakis, C. (2018). Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions. Journal of Cybersecurity, 4(1). https://doi.org/10.1093/cybsec/tyy001

[17] Amram, D. (2020). Building up the "Accountable Ulysses" model. The impact of GDPR and national implementations, ethics, and health-data research: Comparative remarks. Computer Law & Security Review, Volume 37, 105413, ISSN 0267-3649, https://doi.org/10.1016/j.clsr.2020.105413.

[18] U. Mustafa, E. Pflugel and N. Philip, "A Novel Privacy Framework for Secure M-Health Applications: The Case of the GDPR," 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3), 2019, pp. 1-9, doi: 10.1109/ICGS3.2019.8688019.

[19] M. A. Sahi et al., "Privacy Preservation in e-Healthcare Environments: State of the Art and Future Directions," in IEEE Access, vol. 6, pp. 464-478, 2018, doi: 10.1109/ACCESS.2017.2767561.

[20] JPC Rodrigues J., de la Torre I., Fernández G., López-Coronado M. (2013). Analysis of the Security and Privacy Requirements of Cloud-Based Electronic Health Records Systems, J Med Internet Res 2013;15(8):e186, DOI: 10.2196/jmir.2494

[21] Shaping Europe's digital future. 2016. Code of Conduct on privacy for mHealth apps has been finalised. Retrieved on May 30, 2022 from `https://digital-strategy.ec.europa.eu/en/library/code-conduct-privacy-mhealth-apps-has-been-finalised`

[22] ARTICLE 29 DATA PROTECTION WORKING PARTY. Guidelines on Data Protection Impact Assessment (DPIA) and Determining whether Processing Is "Likely to Result in A High Risk" for the Purposes of Regulation 2016/679. Retrieved on May 31, 2022 from `https://ec.europa.eu/newsroom/article29/items/611236`

[23] Georgiou, D., & Lambrinoudakis, C. (2021). Data Protection Impact Assessment (DPIA) for Cloud-Based Health Organizations. Future Internet, 13(3), 66.

[24] French Data Protection Authority Privacy Impact Assessment (PIA). Retrieved on May 31, 2022 from `https://www.cnil.fr/en/privacy-impact-assessment-pia`

[25] Art. 5 GDPR Principles Relating to Processing of Personal Data. Retrieved on May 31, 2022 from `https://gdpr-info.eu/art-5-gdpr/`

Distributed Transactions in Microservices

Radu M. Doroș
Universiteit van Amsterdam
1012 WX Amsterdam, Netherlands
radu.doros@student.uva.nl

Alina Boshchenko
Universiteit van Amsterdam
1012 WX Amsterdam, Netherlands
alina.boshchenko@student.uva.nl

Igor Mazurek
Universiteit van Amsterdam
1012 WX Amsterdam, Netherlands
igor.mazurek@student.uva.nl

Abstract

In this article we explore the design options systems have at their disposals when they need to support transactions spanning multiple services in the context of a microservices architecture. We explore based on the desired consistency levels, what are the common tools systems base their design on.

1 Methodology

In this section, we explain the methodology defined to allow an in-depth analysis of the state of distributed transactions in microservice architectures. For conducting a literature study. we opted to start with the collection of information from the literature. We analyzed around 35 papers from digital libraries, such as ResearchGate, Scopus and Scholar, read several blog posts and tutorials from cloud providers such as Amazon Web Services, Google Cloud and Microsoft Azure, and found N resources that met our selection criteria.

2 Introduction

2.1 Microservices Advantages

Microservices are an ubiquitous way of designing systems in today's software organizations. [24] provides an overview of both advantages and pain points of using miscroservices. Among the most mentioned benefits we can highlight:

- Technology Heterogeneity: Because with micro-services, systems are composed of many smaller services collaborating together it is possible to have services being developed in different languages and using different technology stacks. Organizations can use this features to experiment with desired emerging languages and technologies in some services before making a full transition organization-wide.
- Robustness: Compared to monolithic applications, microservices will typically fail locally with some correlated services failing, while the rest of the system will be able to still be up.
- Simplified deployments: A similar advantage to previous one is that services can be updated and redeployed individually without necessary affecting other services globally. Newly deployed services and the changes they bring will mostly affect local services that collaborate with them.

- Targeted Scaling: Compared to monolithic applications, scaling with microservices can be more targeted. For example systems can choose which services encounter increased load and can add more replicas of them, something impossible in the monolithic world.
- Simplified team organization and code ownership

2.2 Microservices Criticism

With microservices being so popular and widespread, critiques also appeared. Among the disadvantages of adopting a microservices architecture we can enumerate:

- Network latency: In monolithic applications, services were simply only objects on a same system, communication between them having a much smaller latency compared to today's remote network calls.
- Monitoring and Troubleshooting: Observing and diagnosing bugs in a distributed setting is more difficult.
- Encapsulated databases poses problems in cases the system needs to interact with multiple databases in the system. Both querying and complex transactions spanning multiple services can become serious hurdles, especially when they also need to follow some guarantees (atomicity, some minimum levels of isolation).
- More complicated CI/CD pipelines: as the software may need coordination among multiple services, deployment and integration pipelines are not as straightforward as in monoliths [4].

3 Distributed transactions types

A distributed transaction is a set of operations on data that is performed across two or more data repositories, in majority of cases - databases. It can be coordinated both across separate nodes connected by a network and multiple databases on a single server [6].

For a distributed transaction to happen, transaction managers coordinate the resources, manager decides whether to commit or rollback a transaction. There could be a single or multiple entities of the transaction manager(s) in a system. It can be one of the data repositories that is to be updated in the course of transaction, or a completely independent resource only responsible for the coordination. In the current context resources are either multiple nodes of a single database or multiple databases.

We can distinguish between two main types of commonly used types of distributed transactions:

- **ACID Distributed Transactions (Atomic, Consistent, Isolated, Durable)**. Extensions of the classic single-node ACID transactions. Systems opting for this type of transactions can not afford to be flexible in their consistency levels. Typical examples for such systems are banking systems.
- **BASE Distributed Transactions (Basically Available, Soft State, Eventually Consistent)**. These kinds of transactions are best suited for systems that don't necessarily depend on increased levels of consistency and are able to explore lower consistency levels to increase availability and/or performance.

In the subsections below we elaborate a bit more in-depth on the differences between these two types of transactions.

3.1 ACID Transactions

Many applications are dependent on their services and databases providing higher degrees of consistency. The question is, how do such systems behave in case they also follow a microservice architecture blueprint and microservices follow the database-per-service style. Commonly, this architecture suffices and performs well in cases data is not required to

be committed across multiple services. Sometimes, however, there is a need to support transactions spanning across services.

ACID transactions guarantee that a database will be in a consistent state after running a group of operations, even in case of unexpected errors. *Atomicity* guarantees that all of the commands in a transaction are treated as a single unit and either succeed or fail together. *Consistency* stands for the fact that changes made within a transaction are fully consistent with database constraints. *Isolation* makes sure that concurrent transactions do not affect each other's outcomes. *Durability* guarantees that, once the database send an event to the client that the data was recorded, the data has in fact been written to a backing store[8].

3.2 BASE Transactions

BASE provides less assurance than ACID, but it is more efficient for scalability and reacts well to the rapid data changes.

Basically Available property means that rather than enforcing immediate consistency, BASE-modelled databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster. *Soft State* property means that data may change over time due to the lack of immediate consistency. The BASE model delegates the responsibility of maintaining consistency to the developer. *Eventually Consistent* denotes that despite BASE does not enforce immediate consistency, it eventually achieves it. Until this state is reached data reads are still possible.

BASE transactions were designed so that system developers can experiment with lower levels of consistency which can be leveraged into improved availability and performance. Similar to the case of single node databases, where lower isolation levels can be used to achieve improved performance, in a distributed setting it is possible to sacrifice consistency levels for performance, and this opportunity provides a big field for exploration.

3.2.1 Related work and literature evaluation

Medjahed at el. (2009) [22] provides the reader with a generalization of ACID properties. According to the authors the properties were designed for traditional applications like banking and may be no longer applicable for large high throughput databases. They propose the properties to be rather seen as: *Recovery* – ability to recover in case of a failure, *Consistency* – this one remains the same, *Visibility* – ability to see intermediate results of other transactions, *Permanence* – ability to save transactions. This generalization allows to relax some of the constraints compared to traditional ACID.

An interesting take on atomicity implementation in e-commerce systems can be seen in the article by Frank and Kofod(2002) [12]. Their take is to use approximated ACID properties in order to reduce the response time caused by locking properties. The transaction pattern uses nested transactions to achieve *approximated atomicity*. Eventually, all of the updates made to the database are either executed or compensated.

Chandra (2012) [13] analyzes the BASE properties of NoSQL databases like Key/value databases, BigTable, and Columnar databases. He uses widely known and used implementations such as Amazon's Dynamo, Google File System, and Hadoop. NoSQL databases that follow BASE design come with many advantages like high scalability and easy deployment. Table 1 highlights the key differences between two types of transaction types according to Chandra. His judgment is based on SQL and NoSQL databases.

Birman et al. (2012) [3] try to overcome the CAP theorem with consistent soft-state replication. Their consistency alternative includes agreement on update and in-memory durability. One could see it as undeniably close to the BASE solution however their approach has a way to have consistently replicated data across the nodes. Another advantage is that solution does not require locks compared to the traditional ACID solution.

ACID [C + A]	BASE [A + P]
Strong consistency	Accomplishes Consistency, Atomicity and Partition tolerance "eventually"
Isolation	Availability first
Focus on "commit"	Best effort
Nested transactions	Approximate answers
Pessimistic: Force consistency t the end of transaction	Optimistic: Accepts temporary database inconsistencies, Eventually Consistent
Suitable for Financial Portals	Suitable for non-financial web-based applications
Safe	Fast
Shared Something (Disk, Memory)	Shared Nothing
Scale UP (limited)	Scale Out (Unlimited)
Simple Code, robust database	Complex code, simple database
Single Machine	A cluster
CA	AP/CA/CP, i.e. any 2 out of 3.
Scale Vertically	Scale Horizontally
SQL	Custom APIs
Full Indexes	Indexing is mostly on Keys
Difficult evolution	Easier evolution

Table 1: ACID versus BASE [13]

4 Distributed transaction management patterns

A distributed transaction can be perceived as an atomic operation that must be synchronized (or provide ACID properties) among multiple participating resources distributed among different physical locations [6]. Unfortunately, for distributed transactions there is no unified perfect solution for all business problems, because the complete ACID model is hard to achieve, maintaining a high level of performance.

The challenge of implementing ACID transactions in a distributed system comes down to consistently resolving transactions across data partitions in as efficient and scalable a manner. Another challenge comes from networking. as with geographically distributed replicas, minimizing cross-region network communication is critical in order to maintain reasonable response latency [1].

In most cases transaction model follows BASE principles, balancing between assurance and scalability/performance. Therefore, in actual applications, the solution would be to choose the approach according to the particular business requirements and characteristics.

When the application is transitioned from monolith to microservices, developers want to avoid system-wide distributed transactions. However, in comparison to monolith, data is distributed more, so there is a need for distributed transactions. Below we describe 3 main architectural patterns, widely used for the distributed transaction management.

4.1 Two-Phase Commit

Two-phase commit is a standardized protocol that ensures that all ACID properties are complete. It is usually considered the most common way to implement transactions. It is imperfect, but the general idea is simple. In this protocol, a coordinator wants to coordinate an operation with all participants:

- Phase 1: The Coordinator broadcasts a prepare message that instructs the participants of the operations they need to perform. Participants answer with their acknowledgements, confirming that they are prepared for the upcoming commit operation.
- Phase 2: The Coordinator sends a commit message if all participants confirmed the previous step. Also avoids committing in case not all the participants confirmed.

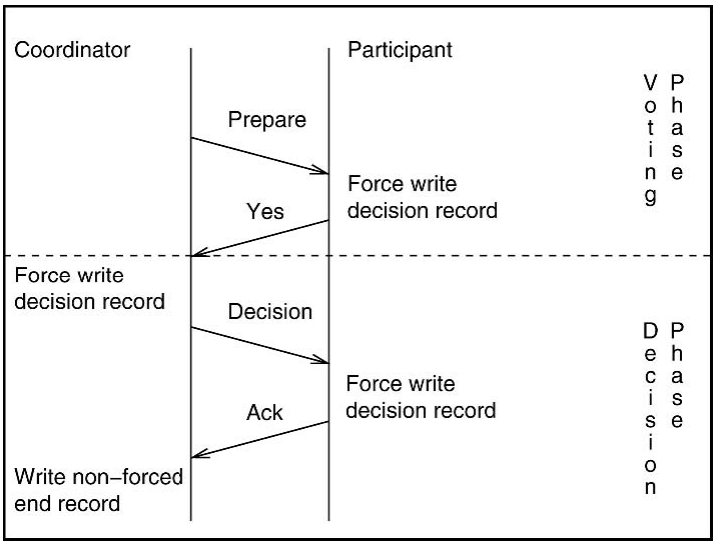


Figure 1: The two-phase commit protocol [18]

Figure 1 presents a sequence diagram of the voting phase and the decision phase. Some of the protocol's main advantages are discussed below:

- The protocol commits the entire transaction or rolls back to its prior state in case of failure in one of the operations.
- It's a simple protocol with much research done since it was published by Lampson in 1979 [21]. According to Al-Houmaily et. al. (2009) [18] it's the most studied *atomic commit protocol*.
- There is a global order which dictates the order in which transactions will be executed. Any updates to the database are isolated, we are not able to see intermediate results of transactions. Simultaneous transactions can occur as long as they do not have an influence on each other.

Unfortunately, 2PC is not perfect that's why we would like to point out some of the problems and design considerations:

- The protocol requires a stable coordinator as it's a single point of failure.
- The protocol is blocking, so if the coordinator fails permanently before sending a message with the final decision, participants don't know whether to commit or abort. This may lead to some transactions never being resolved.
- The more participants are in system, the higher the latency of a transaction. Coordinator needs to communicate with every single participant whenever distributed transaction decisions are made.

According to Samaras el at. (1999)[28], the greatest performance repercussions of the protocol are caused by:

- **Logging.** Logs are helpful in a case of a memory-related system failure, it allows the protocol to reach a consistent state by reading data from stable storage. One could minimalize the number of data saved which would lead to higher performance in non-failure situations. However, recovery time would increase or the independent recovery might be not available at all.
- **Network traffic.** Every message that needs to be exchanged between a coordinator and participants slows down protocol significantly by adding network delay, easiest solution to this obstacle is to limit the number of sent messages in the protocol.

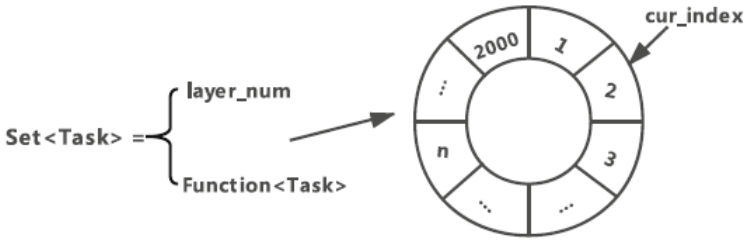


Figure 2: The design of CMQ [10]

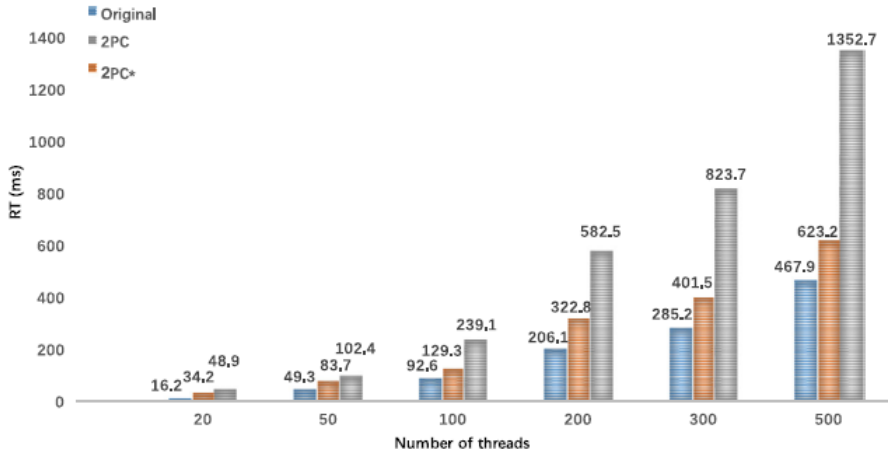


Figure 3: Response time experiment result [10]

4.1.1 Related work and literature evaluation

A substantial number of the research articles regarding the two-phase commit protocol are discussing optimization and possible improvements. One of the most important studies related to the two-phase commit protocol is a paper by Lampson (1979)[21]. It introduces the reader to the protocol through a series of abstractions and informal correctness proof.

The authors of the article *2PC*: a distributed transaction concurrency control protocol of multi-microservice based on cloud computing platform* [10] propose an enhanced two-phase commit with a focus on large-scale and high-throughput systems. It's a innovative concurrency control protocol, by using transaction compensation it tries to improve the fault-tolerance. One improvement is the usage of a circle message queue(CMQ), presented in Figure 2. This mechanism asynchronously polls the response until it gets a successful response, instead of blocking the whole system. To test and evaluate the algorithm authors deploy it using the Netty framework on the PaaS cloud platform. Most of the positive impact can be observed with the number of concurrent threads higher than 100. They were able to achieve a 70% improvement in throughput(transactions per second). Given 300 concurrent requests the original 2PC has higher latency by 2.9 times, moreover, the committing rate would be only one-third compared to the modified version. The results of the response time experiment are presented in Figure 3.

Nouali et. al. (2005)[25] extend the protocol to be fitting the mobile wireless environment. The weaknesses are caused by limited computing resources for mobile devices. Moreover, wireless connection means lower bandwidth, high latency, and is much more error-prone. The protocol is able to counter it by adding an additional phase 0 to the protocol that contains commit messages along with logs. As the connection in a mobile environment can be unreliable the coordinator waits for client acknowledgment before removing logs and forgetting about the transaction. Due to protocol not being redesigned, key ingredients stay the same as in the classic two-phase commit. This makes the adoption of any improvement proposal for classic variation also applicable.

Lapmson and Lomet (1993)[20] propose a lightweight version of 2PC, lower cost of messages could be beneficial for the environment where logging cost is high. We can achieve it by reducing the number of saved information this would result in the disability to know which transactions were active before the coordinator crashed. This means that ability of independent recovery could be lost, another disadvantage is losing the transactions that were started but not committed.

An improvement to the availability of 2PC types of systems: have regions using Raft [26] and their leaders participate in a 2PC famously used by Google Spanner [7]. Google spanner uses Paxos groups, but they can easily be replaced by Raft groups, Raft was not yet published then. Figure 4 shows how Google Spanner's organization could be used to design Distributed Transactions in microservices.

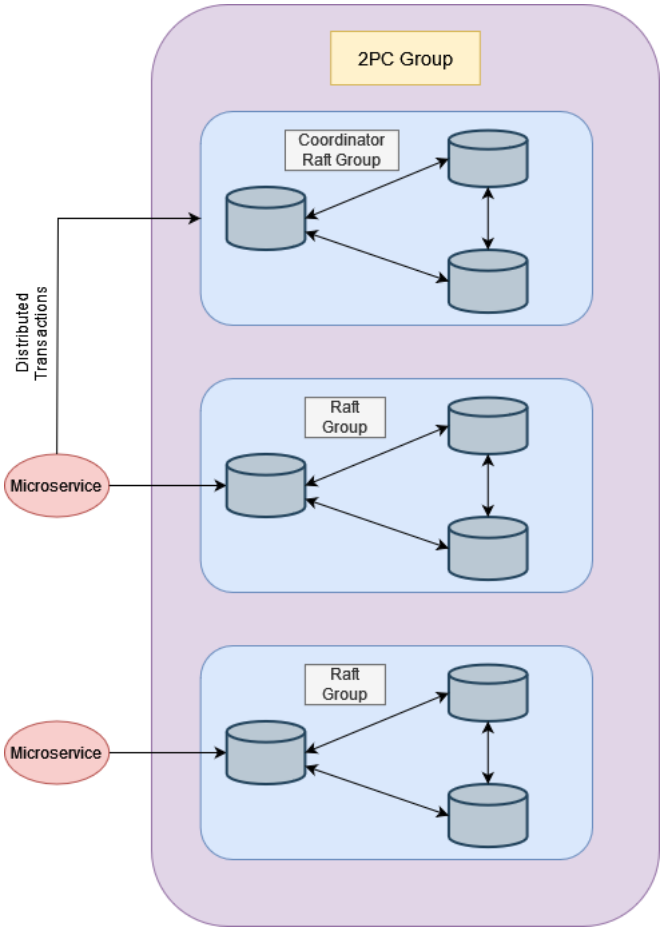


Figure 4: Spanner organization ported to microservices

A workaround developers can sometimes choose is to not opt for the database-per-service style and use a shared distributed DB that will handle such hurdles by itself. For example systems can be designed instead of database-per-service as table-per-service, that also offers to some degree some of the benefits of isolation gained by database-per-service.

One of the first papers (Helland 2007) that explore methods of avoiding distributed transactions in highly-scaling systems from a more architectural perspective is [17]. Helland claims that the benefits 2PC brings are much outweighed by their costs: namely the availability and performance problems. The Author splits layers of micro-service system in two parts: scale-aware and scale-agnostic and argues for simplified designs where transactions have only local scopes and services can run atomic transaction only across local entities. He continues by observing that as systems grow in scale, are more likely related entities spread around

and systems need to repartition their data when this happens. Details and impacts of these repartitions are not discussed in depth, even though they are of high interest.

He goes on and describes ways to correlate transactions across services using *workflows* and *activities*. The operations the author is describing closely resemble Sagas, the paper is published at a time when Sagas did not enjoy same popularity they do today. The description of the operations and concepts offer important insights into how one should design saga their operations.

4.2 Event Sourcing

Most often applications use a traditional model to store and interact with persistent storage called *CRUD*. This acronym stands for four basic operations that can be executed on resources namely - create, read, update and delete. It allows users to update the current state by modifying values in a database. This approach has some weaknesses - limited scalability, and performance as changes as committed directly to the data store. Considering multiple users sharing one resource, complete data loss is possible by other user actions like overwriting or deleting. Additionally, version control is not possible unless there is a separate log implemented. One solution to the above-mentioned problems could be a pattern called *Event Sourcing*.

4.2.1 Concept

Event sourcing means that instead of only keeping conventional mutable states in databases, services append to an immutable log of events. These events are being in turn handled and replayed by services, replaying these events allows the consuming services to be able to reconstitute the current state of an entity's value. Events are appended to a store of events like in Figure 5. The Service issues change events and when all events are replayed by interested services, they will reconstitute the new current state

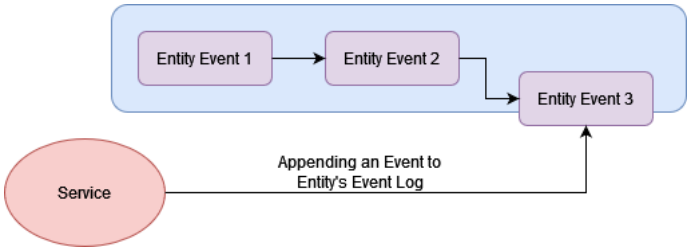


Figure 5: Service appends event change the current of some entity.

Events represent changes to values of an entity's field. Then, a microservice interested in the state of this entity will replay the series of events for this entity. After the replaying, the current value of the entity's fields will be computed.

One of most popular ways to implement the Event Sourcing architectural pattern is using Apache's *Kafka* [19] .

Because the number of events can grow large in some cases, making replaying the full set of events very costly, some ideas are used to avoid long replays. *Snapshotting* and *CQRS* [11] are often used in combination with Event sourcing. Snapshotting is a performance optimization and means saving states of entities after a certain number of events are appended to an entity's event log to avoid following services to replay the full event history.

Usage of CQRS consists in segregating the Commands query model (queries that modify data) and the Read Query model in such a way to allow systems to scale these two types of queries separately. Typically Read Query is separated by replaying events and storing these values in traditional databases that handle Querying well and have a powerful querying language. As an example for a system following CQRS, we could have services performing command queries by using an Event Store, where they can pusblish new events to modify some entity's state and in the background, some services can listen to these events and replay

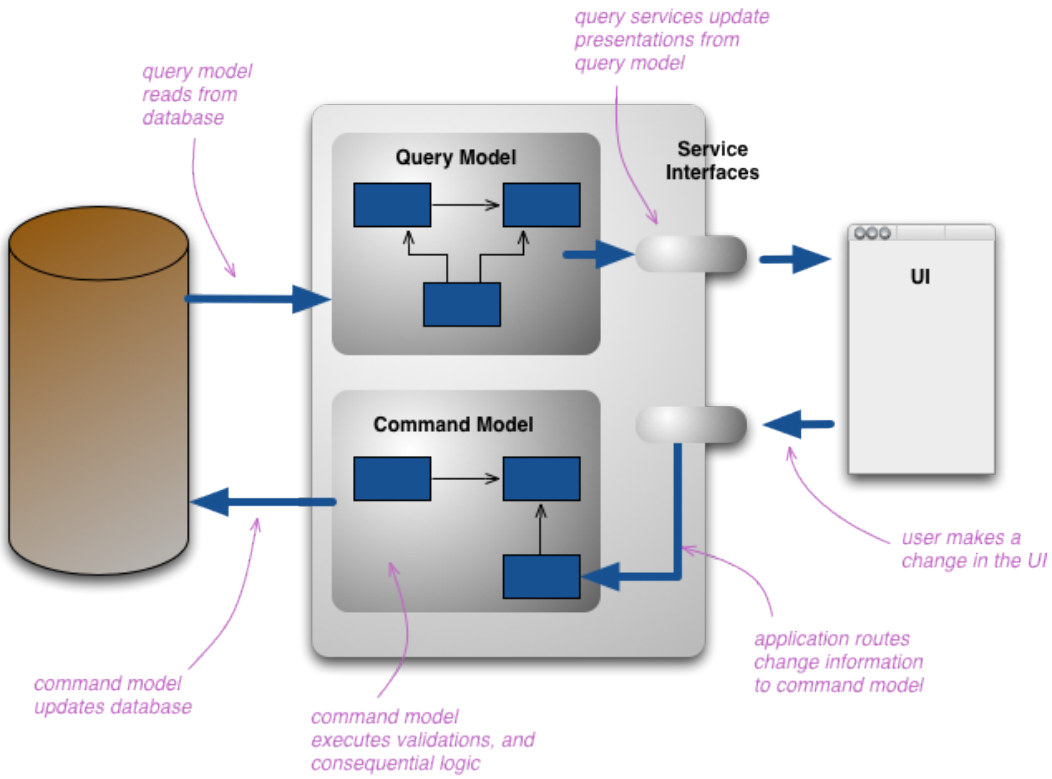


Figure 6: CQRS Architectural Pattern [11]. Typically the database on the left is an EventStore and the Query Model consists of specialized databases

them to insert to a specialized database, for example *ElasticSearch* [15], that handles text finding queries more efficiently. This Query Model will be always somewhat in the past, eventually consistent with the Command Model's state, but this guarantee is most often enough for the typical modern applications. Figure 6 illustrates how CQRS decouples Read Model from Write Model.

Details about database encapsulations in microservices are explored in [16]. Helland categorizes databases of services as *inside* and *outside* data. Inside data denotes encapsulated data contained inside a service itself, while outside data is information that is queryable from outer services, services different than those where the data resides. In the paper's architectural framework, only the service that owns the database is allowed to change data and it will *publish* data to be queried by the outside services. Moreover, from outside services, this published data will be something "from the past". The author distinguishes between the following *reference data* types that are typically encountered:

- *Operands* consist of information published by a service to be used by some other services that perform an *operator*'s job. Services running operators combines the published operands to run their job and possibly produce some results in the system.
- *Historic Artifacts* that report information about what happened in the (distant) past. Many systems need to collect immense amount of data and archive for future statistics and planning (e.g. user's UI interaction patterns, traffic statistics..)
- *Shared Collections* where reference data is shared by multiple services that also want to mutate it. The publishing service also periodically pushes updates. Write conflicts and stale reads, all need to be handled by the clients of this database.

He goes on to explore facets of data encapsulated by their services: storing data from ouside and exensibility. All in all, the article provides an intuitive vision about how microservices interact with data from its own database and with data residing in other microservices and

is especially useful for developers designing systems that follow the CQRS architectural pattern.

4.3 Sagas

4.3.1 Concept

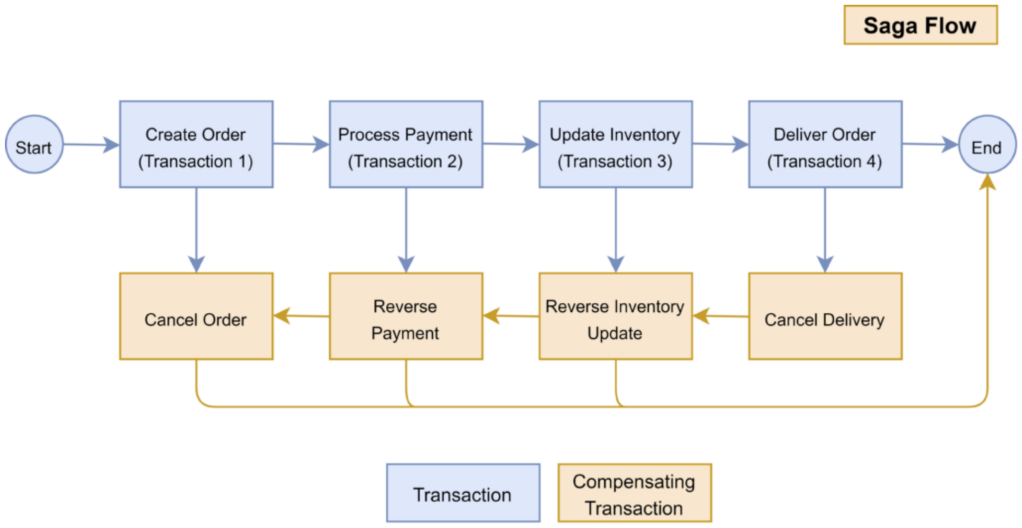


Figure 7: Saga Workflow

The concept of saga was first introduced in a paper in 1987 [14], and the concept gained a lot of attention in the context of microservices [6]. Sagas are similar to nested transactions. A nested transaction is a transaction started by an instruction within the scope of the transaction which is already in progress. In sagas, each of these transactions has a corresponding **compensating transaction**. The reason behind it is the following: if a transactions in a saga fails, the compensating transactions for each transaction that was successfully run previously will be invoked to counteract the preceding transactions. Figure 8 illustrates this behavior.

Due to this logic, the saga guarantees that basically only two states are possible: either all operations complete successfully or the corresponding compensation actions are run for all executed operations to cancel the partial processing.

An **operation** denotes to a part of the saga, which represents a particular work segment. Each saga can be divided into a sequence of operations which can be implemented as a separate ACID transactions. On the operation complete state, results are persisted in the durable storage meaning that it may cause an inconsistent state between the individual operations invocations. However, the saga utilizes the eventual consistency model, which denotes that the state will become eventually consistent after the whole saga completes (both successfully or by compensations calls)[29].

According to the pattern, a compensating transaction must be retryable and idempotent. These principals are strongly required for a transaction to be managed without any manual intervention. The **Saga Execution Coordinator (SEC)** ensures that these principles are hold.

The **Saga Execution Coordinator** is the central component of the Saga pattern. It contains a **Saga log**, which contains all the transactions-related events recordings and can be inspected in cases of failures to determine the operations impacted. In case of SEC failure, it is possible for it to read Saga log once it's coming back up and take all the corresponding actions depending on the transactions' states.

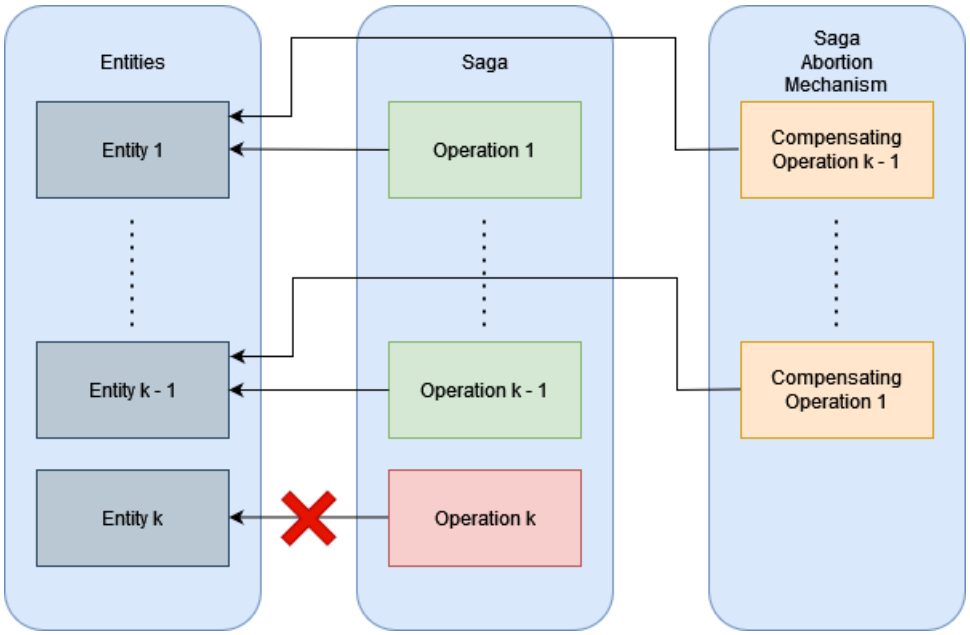


Figure 8: Sagas Compensating Operations

In contrast to the traditional transaction approach and patterns like Two-Phase Commit Protocol, the Saga pattern relaxes the ACID requirements to opt for scalability and availability. Saga breaks the isolation property as it commits each operation separately, which results in a fact that updates of the not fully committed saga are immediately visible to other parallel operations. So, instead of using ACID model, Saga follows BASE model described above in a section 3.2.

Let us now describe the profits of utilizing Sagas in the distributed systems. Saga definition in distributed systems is redefined as a sequence of requests which are placed on a particular participants invocations. The ability of these requests to provide ACID guarantees is not restricted and must be ensured by individual participants.

Saga Execution Component (distributed and durable) and Saga Log also present in distributed systems. Exactly like in the centralized system, each participant is required to expose the idempotent compensating request handler. This handler is capable of semantically undoing the participant's request in the saga.

The main challenge that arises in this case would be the network latency and participant failures that may happen between remote invocations.

4.3.2 Implementation

There are two approaches [27] to implement the Saga pattern, which we describe below.

- **Choreography-based.** Using the EventStore we are able to leverage the existing event handling communication in the system and define saga flows using them.

  In this pattern each microservices that is part of the transaction publishes an event that is processed by the next microservice [2]. The main challenge here is to decide whether the microservice is going to participate in Saga. In this pattern, the Saga Execution Coordinator is either embedded within the microservice or is a standalone component.

  The flow is considered successful if all the microservices complete their local transaction meaning there is no failure reported by any of the microservice. However, if failure occurs, the microservice reports to the Saga Execution Coordinator, and it invokes the relevant compensation transactions. Pattern's behavior is illustrated in Figure 9

The Choreography pattern is suitable for the microservice application with development being is on the early stage. Also, it is a good fit when there are fewer participants in the transaction. It doesn't require additional service implementation and doesn't introduce a single point of failure, since the responsibilities are distributed. Among disadvantages of the pattern we can mention potentially confusing workflows if new steps are added, and risk of cyclic dependency between saga participants because they have to consume each other's commands [23].

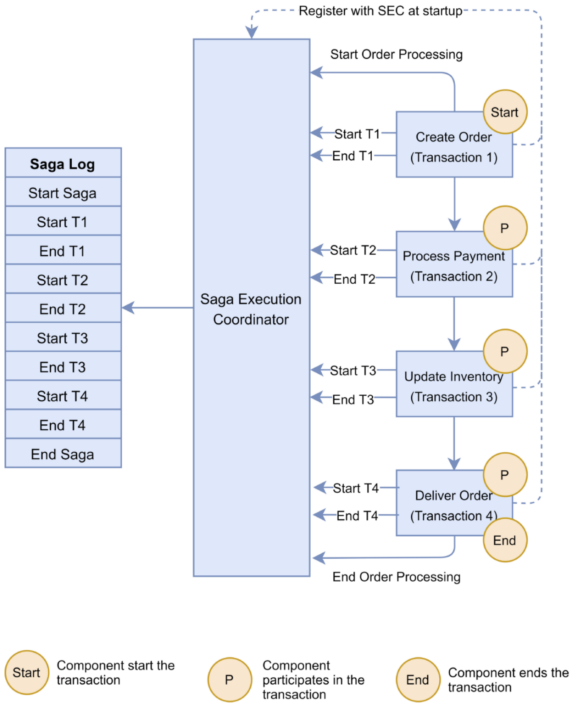


Figure 9: Choreography-Based Saga

- **Orchestration-based.** This pattern uses a central coordinator that orchestrates each participant's actions, it is responsible for managing the overall transaction status. The central coordinator can be a service in the system of microservices that coordinates other services. If any of the microservice encounters a failure, then the orchestrator is responsible for invoking the necessary compensating transactions. Orchestration pattern becomes extremely useful for microservice application on a later development stages. It is good for cases where there are complex workflows involving many participants. It doesn't introduce cyclical dependencies, as the orchestrator unilaterally depends on the saga participants[23]. However, it has higher complexity which requires an implementation of a coordination logic and provides an additional point of failure. This pattern is shown in Figure 10

4.3.3 Discussion.

Usage. Saga pattern is a good fit if it is required to ensure data consistency in a distributed system without tight coupling. It is less suitable for cyclic dependencies and tightly coupled transactions.

Challenges. Saga pattern can be difficult to implement and hard to debug, the complexity grows as participants increase. It is not possible to roll back a transaction as the participants commit changes to their local databases.

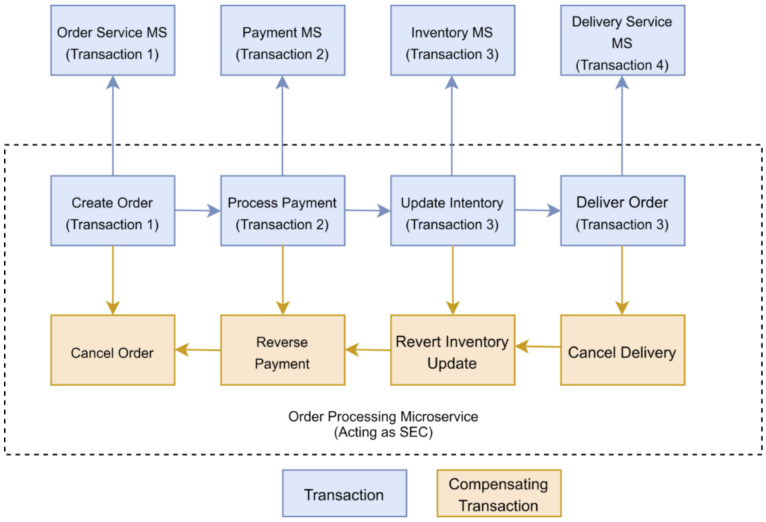


Figure 10: Orchestration-Based Saga

Best practices. It's best to implement monitoring to track the saga workflow. The lack of participant data isolation imposes durability challenges. The saga implementation must include countermeasures to reduce anomalies [23].

4.3.4 Related work and literature evaluation

We found a sufficient amount of research papers, articles and tutorials to understand the concept of Saga, its main strength and weaknesses. Among all the scientific papers we can highlight [9], [5], [30], which clearly define main strength and weaknesses of the pattern, however, these articles do not provide any implementation insides. Therefore we referred to tutorials, in particular, [23] and [2] useful for deeper understanding of the implementation details.

5 Literature evaluation

After analysing the relevant resources for this literature study we came up with the following general overview of the field. Below we present what we enjoyed and found appealing:

- A lot of resources and best practices guidelines about tansactions overall and transactional patterns in monolith applications.
- A decent amount of modern research papers regarding Saga pattern, published after 2018-2019 meaning that the technology holds the attention.
- Clear definition of key differences and strong/weak sides of 2PC and Saga as of patterns implementing the opposite policies.

Understandably not every aspect of such a broad and interesting field was appreciated by us, below we present some of the things that were a bit off-putting:

- Less resources and best practices guidelines for distributed applications.
- Lack of research regarding the future of this topic and discussions on how existing strategies can be enhanced.
- Controversial opinions regarding 2PC

6 Conclusions

Generally, in a microservice architecture, a distributed transaction is an outdated approach that can cause severe scalability issues. Modern patterns that rely on asynchronous data replication or model distributed write operations as SAGAs avoid these problems. 2PC distributed transactions are considered a bad practice, they create a single point of failure and also overload the transaction coordinators heavily. Improvements try to mitigate some of the issues. Sagas, despite all benefits, can become complex to implement especially Choreography-based ones. They are difficult to follow and log. To do this one needs to keep track of full flow of event messages in the EventStore and engineers need to have a good grasp of the the system as a whole.

References

[1] Daniel Abadi and Matt Freels. "Achieving ACID Transactions in a Globally Distributed Database". In: (2017).

[2] Baeldung. *Saga Pattern in Microservices*. https://www.baeldung.com/cs/saga-pattern-microservices. 2022.

[3] Kenneth P Birman et al. "Overcoming cap with consistent soft-state replication". In: *Computer* 45.2 (2012), pp. 50–58.

[4] Lianping Chen. "Microservices: Architecting for Continuous Delivery and DevOps". In: *ACM Sigmod Record* (2018).

[5] Binildas Christudas. "Deployment and communication patterns in microservice architectures: A systematic literature review". In: *Journal of Systems and Software 180* (2021).

[6] Binildas Christudas. *Practical Microservices Architectural Patterns*. 2019. Chap. 6. Distributed Messaging.

[7] James C. Corbett et al. "Spanner: Google's Globally Distributed Database". In: *ACM Trans. Comput. Syst.* 31.3 (Aug. 2013). ISSN: 0734-2071. DOI: 10.1145/2491245. URL: https://doi.org/10.1145/2491245.

[8] MongoDB documentation. *What are ACID Transactions?* https://mongodb.com/basics/acid-transactions. 2021.

[9] Karolin Dürr and Guido Lichtenthaeler Wirtz. "Patterns for Microservices-Centric Applications". In: *13th Central European Workshop on Services and their Composition* (2021).

[10] Pan Fan et al. "2PC*: a distributed transaction concurrency control protocol of multi-microservice based on cloud computing platform". In: *Journal of Cloud Computing* 9.1 (2020), pp. 1–22.

[11] Martin Fowler. "Cqrs". In: *Martin Fowler's Blog* (2011).

[12] Lars Frank and Uffe Kofod. "Atomicity implementation in e-commerce systems". In: *Proc of the Second International Conference on Electronic Commerce, ICEB*. 2002.

[13] Deka Ganesh Chandra. "BASE analysis of NoSQL database". In: *Future Generation Computer Systems* 52 (2015). Special Section: Cloud Computing: Security, Privacy and Practice, pp. 13–21. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0167739X15001788.

[14] Hector Garcia-Molina and Kenneth Salem. "Sagas". In: *ACM Sigmod Record* 16.3 (1987), pp. 249–259.

[15] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.", 2015.

[16] Pat Helland. "Data on the Outside vs. Data on the Inside: Data kept outside SQL has different characteristics from data kept inside." In: *Queue* 18.3 (2020), pp. 43–60.

[17] Pat Helland. "Life beyond distributed transactions: an apostate's opinion". In: *Queue* 14.5 (2016), pp. 69–98.

[18] Yousef J Al-Houmaily and George Samaras. *Two-Phase Commit*. 2009.

[19] Jay Kreps, Neha Narkhede, Jun Rao, et al. "Kafka: A distributed messaging system for log processing". In: *Proceedings of the NetDB*. Vol. 11. 2011, pp. 1–7.

[20] Butler Lampson and David Lomet. "A new presumed commit optimization for two phase commit". In: *19th International Conference on Very Large Data Bases (VLDB'93)*. 1993, pp. 630–640.

[21] Butler Lampson and Howard Sturgis. "Crash Recovery in a Distributed Data Storage System". In: *Unpublished technical report, Xerox Palo Alto Research Center* (June 1979).

[22] Brahim Medjahed, Mourad Ouzzani, and Ahmed Elmagarmid. "Generalization of acid properties". In: (2009).

[23] Microsoft. *Saga distributed transactions pattern, Azure official documentation*. https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/saga/saga. 2022.

[24] Sam Newman. *Building microservices*. " O'Reilly Media, Inc.", 2021.

[25] Nadia Nouali, Anne Doucet, and Habiba Drias. "A two-phase commit protocol for mobile wireless environment". In: *Proceedings of the 16th Australasian database conference-Volume 39*. 2005, pp. 135–143.

[26] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm". In: *2014 USENIX Annual Technical Conference (Usenix ATC 14)*. 2014, pp. 305–319.

[27] Eugen Paraschiv. "Saga Pattern in Microservices". In: (2022).

[28] George Samaras et al. "Two-phase commit optimizations in a commercial distributed environment". In: *Distributed and Parallel Databases* 3.4 (1995), pp. 325–360.

[29] Martin Stefanko, Ondrej Chaloupka, and Bruno Ross. "The Saga Pattern in a Reactive Microservices Environment". In: *14th International Conference on Software Technologies* (2019).

[30] Chellammal Surianarayanan, Gopinath Ganapathy, and Pethuru Raj Chelliah. "Patterns for Microservices-Centric Applications". In: *Essentials of Microservices Architecture* (2019).

A Comparative Study of the State of the Art Tools for Cloud Monitoring

Sklavos, Antonios
a.sklavos@student.vu.nl

Kyriazopoulos, Dionysios
d.kyriazopoulos@student.vu.nl

Nadif, Amin
a.nadif@student.vu.nl

Abstract

Monitoring cloud-deployed applications, as well as infrastructures, is essential due to the inherent complexity of those systems. Detecting failures and bottlenecks originating in the infrastructure itself is paramount for optimal performance. The specialized monitoring software is complex and often platform-specific. Based on our bibliography and former research we present the state-of-the-art of cloud monitoring tools. We set multiple evaluation dimensions and we explain them clearly. We compare and contrast the tools based on those evaluations. Finally, we draw conclusions and suggest fields of further research.

1 Introduction

Back in the 00s, cloud computing was a revolutionary term, a mere suggestion that would transform the IT industry from its core. Today, cloud computing has fulfilled its potential, revolutionized the industry, and become a mainstay in almost every kind of business. The NIST (National Institute of Standards and Technology)[24] definition of cloud computing: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. NIST also acknowledges five essential characteristics of cloud computing which are: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. This cloud model is composed of three service models which are Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (Paas). Furthermore, it describes private cloud, public cloud, community cloud, and hybrid cloud as its deployment models. In layman's terms, Rashid et al.[12] explain that cloud computing essentially replaces our computer's hard drive with the Internet. Instead of storing and accessing data to and from our hard drives, we perform the same actions through the internet.

This literature review aims to introduce and describe the term cloud monitoring, present the state-of-the-art tools, discuss their similarities and differences and finally provide ideas for further research.

1.1 Cloud Monitoring

The cloud is a large, distributed system that considers virtualization technology to manage resources. Availability, concurrency, dynamic load balancing, independence of running applications, security, and intensiveness are all needed to provide a cloud system. Along with these attributes, Cloud computing must deal with challenges as well. These challenges include quality of service (QoS), provision and guarantee of SLAs (Service-Level Agreement), and management of large-scale, complex, and federated infrastructure. To face these challenges, accurate and fine-grained monitoring activities are required.

1.1.1 Need for Cloud Monitoring

Ward et al.[30] break down the motivation for cloud monitoring into five essential pieces. We briefly explain those pieces.

- **Performance**: Find ways to investigate the performance jitter, detect resource over-sharing, select ideal instances according to the user's needs and financial capabilities, and determine a performance benchmark for future deployments.
- **SLA enforcement**: Notice unusual SLA violations like high error rates in APIs and performance degradation of VMs. Ensure safeguarding performance for the user.
- **Load balancing latency**: Monitor load balancing to avoid incorrect traffic distribution and to ensure the creation of additional VMs when required.
- **Service faults**: Ensure service availability and correctness, specifically when an entire business depends on one cloud provider.
- **Location**: Decide the geographic location of a VM based on all relevant information. Ensure that the VM is placed as close to the user or the data source according to the user's demands.

In another approach, Aceto et al.[14] emphasize the importance of cloud monitoring for both the Cloud Service Providers (CSP) and the Cloud Service Consumers (CSC). Cloud monitoring provides essential information, which should serve as assurance to the SLAs and the quality of service.

Syed et al.[29] move one step forward and describe both the Cloud Service Providers and Cloud Service Consumers' perspectives. The CSPs consider efficient resource utilization an important factor, both performance-wise and financially-wise. They also want insurance regarding the SLA boundaries of the service level that the user anticipates. Finally, monitoring physical infrastructure is paramount in avoiding service interruption and machine failure. On the other hand, CSCs are mainly concerned about the promised service level and the QoS. Additionally, they need to monitor the condition and health of the virtual machine at any second.

From another perspective, Hauser et al.[20] pick three major categories, to divide the motivation for cloud monitoring. The first category is alerting, which includes ensuring service availability, detecting errors and failures, and preserving the health of the system. This can be likened to Ward's[30] service fault category. Secondly, resource allocation aims to monitor load balancing and avoid incorrect traffic distribution or the creation of VMs when not required. This category is likened to Ward's load balancing latency category. Finally, visualization of current and historic measurements could help in controlling an enterprise's resources. We observe that every approach has significant similarities in motivating cloud monitoring.

1.1.2 Cloud Monitoring Taxonomy

Cloud Computing involves many activities for which monitoring is an essential task. There exist different perspectives[13][14][29] regarding this taxonomy of essential tasks. After carefully investigating them, we produced the following taxonomy:

- **Capacity and Resource Planning**: Developers have to investigate the capacity and the resources, measure their performance, and predict their demand. They also need to consider how such applications and services are designed and implemented. Finally, they need to estimate the workload.
- **Capacity and Resource Management**: Avoid faults and errors when migrating virtual resources from one physical machine to another. Maintain 100% uptime and provide high availability.
- **Data Center Management**: Cloud services revolve around huge data centers and utilize large-scale data engineering. To keep these data centers running smoothly, developers have to monitor the hardware and provide essential metrics. Furthermore, they have to analyze the data and apply the results to resource provisioning or troubleshooting actions. Paramount to achieving this, is the scalability and the real-time operation of these actions.
- **SLA Management**: Monitor SLA parameters to avoid violations and subsequently financial or legal actions. Moreover, monitoring SLA parameters allow the developers to create better

pricing models, by creating more realistic SLAs based on the performance shown to the user.

- **Billing**: Cloud computing exploits the model of "utility computing". Consumers pay proportionally for the use of the service, according to the type of price model and the type of resources they have chosen. Monitoring is mandatory to assure the consumer that he is paying for what he is using. It is also important for the developer to measure the billing in order to generate accurate and verifiable bills.
- **Troubleshooting**: A monitoring system is an important tool for investigating faults and identifying where the error exists or which is the underlying problem. The system should be highly reliable and available to ensure that the client knows the reason for the failure in a timely manner.
- **Performance Management**: Real-time applications are dependent on the cloud's high availability. For users to select the most applicable cloud according to their needs, monitoring the cloud's performance is crucial. Monitoring the cloud's performance and metrics can allow the user to switch clouds in real-time with confidence and ease. Performance monitoring also ensures that the SLAs are not violated.
- **Security Management**: Critical services and agencies use clouds daily. Highly precise and strict security monitoring is mandatory to guard sensitive information and satisfy relevant regulations and rules.

Cloud monitoring is paramount for each one of the aforementioned fields. It is responsible for providing accurate and precise metrics and statistics and helping cloud developers deal with any kind of danger.

1.1.3 Cloud Monitoring Properties

A cloud monitoring system must have specific qualities in order to be successful. These properties are analyzed by Bulla et al.[17] and others [14],[13]. In this subsection, we briefly present the properties that are needed for the smooth operation of a cloud monitoring system.

1. **Accuracy**: A cloud monitoring system is accurate when it provides measures that are as close as possible to the real value. Cloud services need accuracy for tasks like troubleshooting and SLA management, where a slight error can be misleading for problem-solving or can lead to a violation of SLAs.
2. **Adaptability**: Requirements, resources, and demands change dynamically in the cloud. A cloud monitoring system is adaptable when it can support this dynamic nature and adapt to varying computational and network loads.
3. **Autonomic System**: Reacting extremely fast to errors and faults of the cloud services is crucial for a monitoring system. Equally important is to avoid human intervention in dealing with these mishappenings. A cloud monitoring system is autonomic when it can react automatically to unpredictable changes without exposing them to the consumers. This property relies upon other properties like timeliness and elasticity.
4. **Availability**: A cloud monitoring system should be available at any point for any user that requests its services. Availability is important for tasks like SLA management, billing, and resource and capacity management.
5. **Comprehensiveness**: A comprehensive cloud monitoring system can support multiple types of resources, either physical or virtualized, multiple types of monitoring data, and multitenancy. Comprehensiveness is important for both providers and developers because it enables adaption to any monitoring API without considering the type of monitoring information.
6. **Elasticity**: Cloud computing needs to be dynamic to cope with the various assignment of resources to users, multiple monitoring requirements, and multitenancy. Thus, a cloud monitoring system is elastic when it deals with dynamic changes, and consequently, it depends on scalability. The monitoring system should be able to monitor correctly, dynamically created, or destroyed virtual resources. Elasticity is also called dynamism.

7. **Extensibility**: Extensibility is the functionality that extends the cloud monitoring systems to match the new demands of each user. An extensible cloud monitoring system provides the opportunity to enhance the comprehensiveness of the system without tampering with the monitoring framework.
8. **Intrusivity**: A cloud monitoring system is intrusive if adaptability means significant modifications to the cloud. When non-cloud monitoring systems extended to the cloud, they retained low intrusiveness along with their extensibility.
9. **Scalability**: A scalable cloud monitoring system is able to deal with the huge volume of data collected by many virtual resources on top of a single physical resource. Consequently, the system can effectively analyze this volume of data without being impaired by neither a large number of parameters nor a large number of resources.
10. **Reliability**: Reliability is paramount for the QoS of cloud services. A cloud monitoring system is reliable when it can do certain actions under specific conditions and for a known period of time.
11. **Resiliency**: A cloud monitoring system is resilient when it can withstand several failures without compromising its operations. Resiliency is a key part of ensuring the smooth operation of tasks like billing and SLA management.
12. **Timeliness**: Timeliness is a complex property. Failure to obtain the correct information on time erases the usefulness of monitoring itself. Timeliness is affected by three core factors: sampling, analysis, and communication delay. Each of these factors can affect the cloud monitoring system positively or negatively, and there is always a trade-off between speed and sampling when requiring up-to-date information. Thus a cloud monitoring system is timely when it detects events on time for their intended use.

Figure 1 provides a schema of the cloud monitoring properties and their interconnections. It also pinpoints the issues each property has to deal with. In certain approaches[17] multitenancy, which means concurrently handling requests and data from multiple users, and portability, which means that cloud environments must work with different platforms and services, are included as well in the cloud monitoring properties.

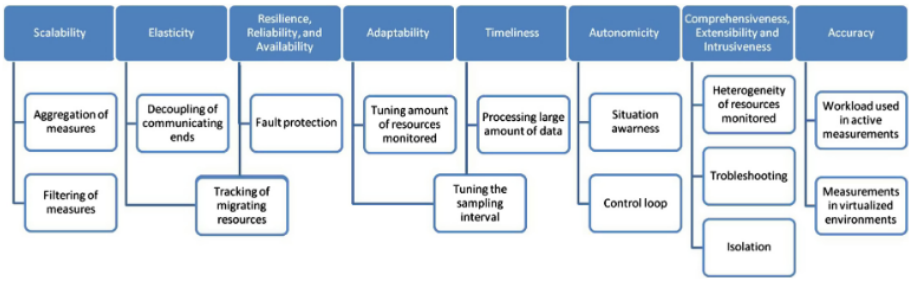


Figure 1: Cloud monitoring properties and related issues

1.1.4 Monitoring by Cloud Layer

Cloud Security Alliance models the cloud into seven layers. Spring[26][27] introduces those layers through the CSA (Cloud Security Alliance) and presents monitoring techniques for each one of them.

The **Facility Layer** requires physical monitoring which, includes surveillance, architectural resiliency, and security guards. Furthermore, the operation of a data center comes with great responsibility and should be accompanied by a comprehensive continuity of operations plan (COOP).

Control and monitoring of the **Network Layer** are essential in securing the operations between the provider and the consumer. Protection mechanisms like firewalls, dynamic firewalls, intrusion detection systems (IDSs), intrusion prevention systems (IPSs), and network proxies are the standard techniques for protecting the cloud's border from external dangers. Collecting and logging information from the monitoring is also useful and can serve as validation for the SLAs.

Securing the **Hardware Layer** begins with monitoring the hardware and ensuring high availability. Measuring values like memory use, bus speeds, processor loads, disk storage, temperature, and voltage can lead to correct and timely load-balancing.

The **Operation System** of a cloud service must be secured and monitored. If the host OS is compromised, then sensitive customer data is compromised as well. If the providers monitoring the OS notice unusual changes, they can return the OS to a known good state. Furthermore, the customer should apply standard practices to monitor his OS.

The security of the **Middleware** is significant in maintaining a cloud provider,Äôs or customer,Äôs information assurance capabilities. Malicious acts against the middleware can be as harmful as against the OS. Techniques to protect the middleware include: maintaining an architecture that is simple enough to prevent errors in the settings, third-party inspection of the developer's source code and reporting to the customer, and finally monitoring of security services if they are provided by the middleware.

The source code and the business logic of each **Application** should be examined by third parties and reported to the customer. Regarding web applications, cloud developers should apply precautions like deploying digital certificates (SSL) and performing domain authentication. Additionally, applications should sanitize all inputs.

The **Users** who are members of the customer organization are important to the security policy. Cloud developers can monitor access patterns to recognize malicious behavior. Furthermore, the education of the user regarding the dangers of the web is an integral part of security.

2 State Of The Art

In this section, we will present state-of-the-art tools for cloud monitoring. We will split these tools into two distinctive categories: Commercial and Open-Source tools. We will briefly describe the majority of the tools from each category and delve deeper into a small selection of them. There will be comparisons based on the already mentioned cloud monitoring properties. Furthermore, we will provide information about each tool's deployment model and monitoring architecture. We will also discuss each tool's service model, wherever applicable.

The National Institute of Standards and Technology (NIST)[24] provides its definitions for each model. The **private cloud** model is provisioned for exclusive use on behalf of a company. Consequently, this model is inaccessible to the public. The private cloud can exist on or off premises and can be operated by a third party. It is considered the most secure deployment model. The **public cloud** infrastructure is designed for use by the general public. A company, a business, a government, or an academic body can operate it. It exists on the premises of the cloud provider. Famous examples of public clouds include the Amazon Elastic Cloud Compute and the Google app engine. The **community cloud** is designed for exclusive use by a specific group of consumers from different organizations with shared interests. As a result, more than one organization can manage it, and it can exist on or off-premises. Finally, the **hybrid cloud** is a composition of two or more distinct cloud infrastructures like the ones mentioned above. Despite being bounded together, these cloud infrastructures remain unique.

There are different research papers[23][15][28] analyzing the monitoring architecture of the cloud monitoring tools. Centralized monitoring architecture and decentralized monitoring architecture are the two distinct categories.

- **Centralized Architecture**: In a centralized architecture, the server resources send status updates to the centralized monitoring server. The system collects states from the resources by polling and pushing. Afterward, the monitoring server is responsible for identifying errors, reporting them, and taking control. Drawbacks of centralized architecture include lack of scalability, suffering from a single point of failure, and lack of computational power to handle loads of monitoring requests.
- **Decentralized Architecture**: Decentralized architecture is on the rise. A monitoring tool configuration is decentralized when none of the system's components are more important than the others. In contrast to centralized architecture, if one component fails, then it does not influence the operability of the other components. This type of monitoring system have, also, scalability improvements over centralized systems.

2.1 Commercial Tools

Numerous approaches have taken place regarding commercial cloud monitoring tools. Alhamazani et al.[15] provide information for a few commercial tools, while also comparing them in terms of monitoring architecture and other evaluation dimensions. Birje et al.[16], Aceto et al.[13] and Fatema et al.[19] describe briefly the commercial tools and compare them in terms of cloud monitoring properties. Syed et al.[29] introduce some tools as well and compare them in terms of monitoring architecture and deployment model. Finally, Fahad et al.[18] provide advantages and drawbacks for each tool. To continue, we present some of the commercial cloud monitoring tools.

2.1.1 Monitis

Monitis is a commercial cloud monitoring tool that started in 2006. Since 2011, Monitis operates under the TeamViewer umbrella after its purchase. Monitis is capable of providing cloud-based agentless IT monitoring solutions. It is able to monitor websites, servers, applications, networks, and cloud virtual instances. Its main focus is Amazon Services but also supports Microsoft Azure, RackSpace, VMware, and Hyper. Monitis is a Software-as-a-Service solution and also provides extendable APIs. Additionally, it supports fully customizable widgets for viewing performance data. Finally, it copes well with most essential tasks except for security management.

2.1.2 LogicMonitor

LogicMonitor[7] is a SaaS-based unified observability and IT operations data collaboration platform for enterprise IT and managed service providers. It was founded in 2008 and partners with third parties like Dell and HP.

LogicMonitor adopts an elastic multi-layer approach and ranges from web servers and databases to the underlying supervisors. It supports visualized environments like VMware and Cloud platforms like Amazon and Eucalyptus. It implements extensibility by providing the users with extensible APIs. Furthermore, it stores data for up to two years and assists the customers 24/7. By using AIOps, it enables troubleshooting and proactive prevention of alarms and issues. LogicMonitor is a collaborative effort of RackSpace and NimSoft.

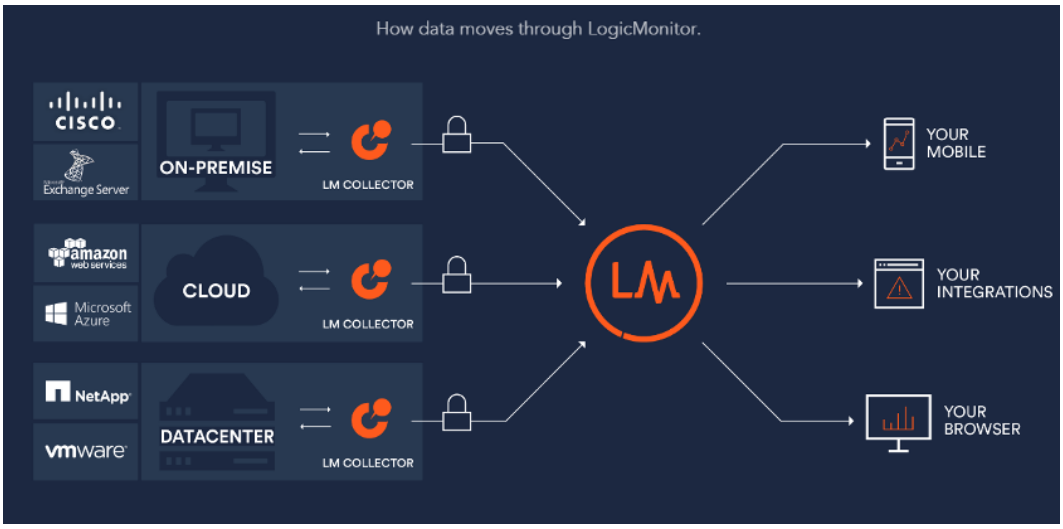


Figure 2: LogicMonitor workflow

2.1.3 CloudMonix

CloudMonix[3] is an enhanced cloud monitoring and automation solution for Microsoft Azure Cloud. CloudMonix replaced AzureWatch, which monitored and aggregated key performance metrics from Azure resources. It provides a live monitoring dashboard that allows the cloud administrators to

explore the cloud resources and get timely information on errors and cautions. CloudMonix supports automatic problem resolution by using AI and scripting, while it can also auto-scale the resources to match the ever-changing demands of the customer. A possible limitation can be that it works only with Azure Cloud resources.

2.1.4 NimSoft

Nimsoft is a cloud monitoring tool able to monitor data centers of both private and public clouds. Nimsoft was bought by CA Technologies in 2010 and it is still used by them. It offers a large variety of IT monitoring, including a plethora of OS servers, storage, and network resources including routers and firewalls. Furthermore, Nimsoft monitors databases like Oracle, applications like Microsoft Exchange, Cloud environments, and VoIP environments by Cisco.

Like LogicMonitor, Nimsoft is capable of extensibility by extending APIs and encouraging custom configuration from the customer. Importantly, it succeeds with each of the essential tasks discussed in the introduction. Nimsoft's drawbacks include non-fault tolerance and not supporting SLA compliance.

2.1.5 CloudKick - RackSpace Cloud Monitoring

CloudKick[9] is a multi-cloud management platform with a range of high and low-level monitoring features and metrics. Cloudkick was acquired by RackSpace, which provides monitoring data like CPU utilization and traffic volume. Through Cloudkick, measured data can be visualized in real-time. Alerts and reports inform the users about errors and cautions.

RackSpace Cloud Monitoring which eventually replaced CloudKick is an effective tool for advanced monitoring configurations and for providing a variety of metric data collection. Cloudkick's disadvantage is that it works only on centralized models and does not ensure availability.

2.1.6 CloudWatch

Amazon CloudWatch[1] is one of the most famous commercial cloud monitoring tools. It is used for monitoring higher-level cloud data. Cloudwatch can store the collected data for up to fifteen months. Customers can build plots, statistics, metrics, and alarms among others, on this data.

A key advantage of CloudWatch is that it allows the customer to collect data from all his Amazon Web Services resources, applications, and services into one single platform. Cloudwatch automatically publishes one-minute metrics with a one-second interval. It can also be used in hybrid environments. Cloudwatch enables consumers to raise alarms and automate actions to identify malicious activities or abnormal behavior. Additionally, automatic dashboards help with the visualization of the data and ensure insight for the users. Consequently, it helps in troubleshooting the user's infrastructure and quickly resolving the root of the problem. Users can create alarms to trigger auto-scaling and, as a result, optimize each action proactively. Furthermore, Cloudwatch collects data from every layer of the performance stack, leading to the monitoring of the client-side data.

In figure 3 provided by Amazon, the workflow of CloudWatch is described.

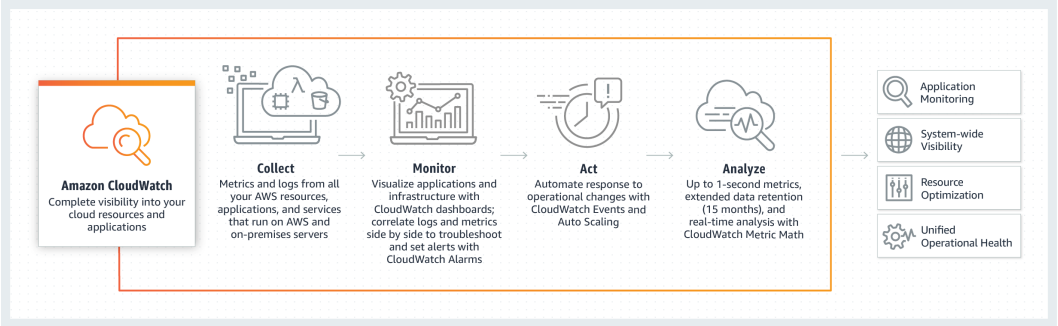


Figure 3: CloudWatch workflow

2.1.7 Comparison

In table 1 the commercial cloud computing tools are compared in terms of key properties. The key properties were extensively described in the introduction1.1.3. Monitis' key properties are comprehensiveness, because it can support various kinds of resources and cloud providers, and extensibility as it provides extendable APIs. LogicMonitor falls in the same category as Monitis and adds elasticity due to its capability to deal with a plethora of assignments and to its elastic multi-layer approach. Because of its automatic recovery procedures, CloudMonix thrives as an autonomic system. In addition, the ability to auto-scale resources in real-time makes scalability its key property. Nimsoft can monitor a broad variety of resources and databases, thus making it comprehensive. Cloudkick-RackSpace's dynamic nature makes it highly adaptable, while CloudWatch's automated responses and auto-scaling leads to timeliness and elasticity.

Table 1: Comparison by key properties

Tools	Accuracy	Adaptability	Autonomic	Availability	Comprehensiveness	Elasticity	Extensibility	Intrusivity	Scalability	Reliability	Resiliency	Timeliness
Monitis	-	-	-	-	✓	-	✓	-	-	-	-	-
LogicMonitor	-	-	-	-	✓	✓	✓	-	-	-	-	-
CloudMonix	-	-	✓	-	-	-	✓	-	✓	-	-	-
NimSoft	-	-	-	-	✓	-	-	-	✓	-	-	-
CloudKick	-	✓	-	-	-	-	-	✓	-	-	-	-
CloudWatch	-	-	-	-	-	✓	✓	-	-	-	-	✓

In table 2 we can observe the comparison of the tools based on architecture and deployment model. Unfortunately, we could not find enough information about the monitoring architecture of the tools. Monitis and Nimsoft work with both centralized and decentralized architecture. On the other hand, CloudKick-RackSpace works only with centralized monitoring architectures. All the tools except for CloudMonix, work with private clouds. CloudMonix works exclusively with hybrid clouds, while LogicMonitor supports this deployment model as well.

Table 2: Comparison by architecture, deployment model

Tools	Centralized	Decentralized	Public	Private	Hybrid	Community
Monitis	✓	✓	-	✓	-	-
LogicMonitor	NA	NA	-	✓	✓	-
CloudMonix	NA	NA	-	-	✓	-
NimSoft	✓	✓	✓	✓	-	-
CloudKick	✓	-	-	✓	-	-
CloudWatch	NA	NA	-	✓	✓	-

Fatema et al.[19] propose a ranking system for both the commercial and the open-source cloud monitoring tools. They score a percentage for each of the tasks proposed in the cloud monitoring taxonomy 1.1.2. We investigate those scores in table 3 and comment on them based on their and our findings. We observe that Monitis' big advantage is its capacity and resource management and planning. The wide range of Monitis is the reason behind this advantage. Its ability to cover most of the IT industry stands out. As mentioned earlier, security is its major disadvantage. LogicMonitor was not part of the Fatema et al. research. We recognize capacity and resource management as LogicMonitor's best task. Its elastic approach and its extensibility plays the biggest part in our reasoning. Additionally, its ability to monitor databases makes up for the data center management category. CloudMonix was known as AzureWatch, when the research took place. We updated the scores based on our study. We emphasized on troubleshooting because CloudMonix's automated solution system matches the demands of the customers. Nimsoft scored the highest in almost every category. We decided to penalise SLA and Security management as they are part of Nimsoft's drawbacks, according to our research. CloudKick provides high and low level monitoring features and metrics. This versatility contemplates for the high score in both capacity and resource planning and management. Finally, we boosted most of the scores for CloudWatch because it deserves it through its longevity and performance.

Table 3: Comparison by taxonomy

Tasks Based on Taxonomy	*Monitis*	*LogicMonitor*	*CloudMonix*	*Nimsoft*	*CloudKick*	*CloudWatch*
Capacity and Resource Planning	82%	81%	80%	91%	87%	82%
Capacity and Resource Management	80%	87%	85%	90%	88%	85%
Data Center Management	65%	85%	NA	87%	NA	NA
SLA Management	81%	89%	81%	80%	88%	81%
Billing	82%	NA	79%	89%	86%	83%
Troubleshooting	77%	85%	88%	83%	80%	83%
Performance Management	NA	NA	NA	NA	NA	NA
Security Management	66%	80%	82%	79%	75%	84%

2.2 Open-Source Tools

2.2.1 Nagios

Nagios Core [8] is a general-purpose open-source monitoring system (Commercial version "Nagios XI" is also offered). It is one of the most popular monitoring solutions on the market. It offers great flexibility through the usage of plugins (compiled executables or scripts). As a result, limitations such as lack of features can be straightforwardly resolved. Both official and third-party plugins exist - Nagios itself does not have internal mechanisms to perform monitoring tasks. [22] Advanced users may benefit from the development of their custom plugins. As a result, it can be adapted to monitor virtually any environment [25]. It works on Linux and Unix variants.

It is especially common for Nagios Core to serve as the basis framework for building more advanced/user-friendly monitoring solutions. [21]

2.2.2 Zabbix

Zabbix [11] is a cloud monitoring solution that started in 2001. It has the ability to monitor servers and services in addition to cloud infrastructures. Contrary to Nagios (2.2.1) it works on macOS, Windows, and Solaris on top of Linux. Zabbix offers easy installation and configuration. [29] Its latest version (6.0 LTS) adds Kubernetes monitoring, offering automatic nodes/pods discovery, and can be deployed on existing enterprise frameworks. A drawback of the framework is that it is unable to scale to support very large environments [29]. Additionally, the auto-discovery feature of Zabbix can be inefficient [19].

Its price and reliability stemming from two decades of operation make it a compelling choice.

2.2.3 collectd

collectd [5] is a Unix daemon, which uses a modular design to minimize the resources it needs [4] in order to collect network metrics - done using plugins.

Its main advantage is its performance, offered by its modular design and pure C implementation. For that reason, it is a popular choice for embedded systems. Additionally, it is actively developed and its documentation is of a reportedly high standard.

A major limitation is its lacking alerting mechanisms (monitoring is limited to simple threshold checking [5]. Additionally, no visualization platform is offered [19]

2.2.4 Ganglia

Ganglia [6] is a popular distributed monitoring system for clusters and Grids. Its main three components are:

- `gmond` (Ganglia Monitoring Daemon)
- `gmetad` (Ganglia Meta Daemon)
- Web front-end

`gmond` must be installed on each cluster that must be monitored. `gmond` collects information about the state of the cluster and sends it to a ganglia meta daemon. Multiple cluster nodes are represented using `gmetad` (point-to-point connections). `gmetad` sends aggregated state information (XML) collected from `gmond` instances, to a client over TCP.

Among its drawbacks, is its inability to be easily customized. Additionally, it introduces network overhead due to the multicast updates, and XML event encoding [19]. Also, Ganglia has no alerting mechanisms.

2.2.5 cacti

Cacti [2] is an open-source network and server monitoring tool, first released in 2001. It is used mainly as a frontend to the industry standard logging software RRDTool [10].

Its advantages include excellent graph support and flexible data sources. In addition, Cacti has a reportedly appealing dashboard with several view options.

Many companies are using Cacti with some other monitoring tools for monitoring their cloud [29].

2.2.6 Comparison

In table 4 we compare the open-source cloud monitoring tools by their properties. To begin with, Zabbix's ability to monitor different kinds of resources makes it comprehensive. It is also known for its reliability and timeliness, due to its auto-discovery feature. Nagios supports the usage of plugins, thus making extensibility one of its key properties. Its higher level of performance ensures accuracy and resiliency. Collectd benefits as well from high performance and consequently it provides high accuracy. Furthermore, it is extendable, through its support for plugins. Cacti's fault management makes it an autonomic system and adds to its resiliency. Ganglia can monitor different kinds of cloud environments making it highly adaptable and scalable.

Table 4: Comparison by properties

Tools	Accuracy	Adaptability	Autonomic	Availability	Comprehensiveness	Elasticity	Extensibility	Intrusivity	Scalability	Reliability	Resiliency	Timeliness
Zabbix	-	-	-	-	✓	-	-	-	-	✓	-	✓
Nagios	✓	-	✓	✓	-	✓	✓	-	✓	-	✓	-
Collectd	✓	-	✓	-	-	-	✓	-	-	-	-	-
Cacti	-	-	✓	-	-	✓	-	✓	-	✓	✓	-
Ganglia	-	✓	-	-	✓	-	-	-	✓	✓	-	-

Table 5 shows each tool's monitoring architecture and support for deployment models. Again, information about the monitoring architecture is limited. Nonetheless, we managed to confirm that Zabbix, Nagios, and Cacti can support both centralized and decentralized monitoring architectures. Furthermore, Zabbix and Cacti can also support both Public and Private cloud models. On the contrary, Collectd and Ganglia can support only private cloud models, while Nagios supports only public cloud models.

Table 5: Comparison by architecture, deployment model

Tools	Centralized	Decentralized	Public	Private	Hybrid	Community
Zabbix	✓	✓	✓	✓	-	-
Nagios	✓	✓	✓	-	-	-
Collectd	NA	NA	-	✓	-	-
Cacti	✓	✓	✓	✓	-	-
Ganglia	NA	NA	-	✓	-	-

Similarly to the comparison section of the commercial tools2.1.7, Fatema et al.[19] also scored the open-source tools based on the tasks of the proposed taxonomy. The first pattern that we notice, is the huge difference in scores between the open-source tools and the commercial ones. As we can see in table 6 the open-source tools have lower percentages in almost every task. The scores that stand out are the ones from Zabbix. We lowered the capacity and resource management score,due to Zabbix's inability to auto-scale when dealing with larger environments. Capacity and resource planning and management for Nagios are high in comparison to the other tools. Nagios' extensibility and adaptability can help in dealing with virtually every environment. Collectd is a rather limited cloud monitoring tool and the scoring depicts that. Average in almost every category and extremely low in terms of security management. Cacti usually accompanies other monitoring tools when used by cloud developers. This limitation leads to its more than average scores. Finally, despite its drawbacks, Ganglia scores more than average and that is reasoned by the work of its three important components.

Table 6: Comparison by taxonomy

Tasks Based on Taxonomy	*Zabbix*	*Nagios*	*Collectd*	*Cacti*	*Ganglia*
Capacity and Resource Planning	87%	75%	66%	56%	73%
Capacity and Resource Management	76%	70%	60%	56%	70%
Data Center Management	NA	NA	NA	NA	NA
SLA Management	88%	66%	56%	47%	59%
Billing	86%	61%	64%	46%	68%
Troubleshooting	80%	57%	60%	43%	63%
Performance Management	NA	NA	NA	NA	NA
Security Management	70%	54%	41%	59%	63%

3 Discussion

3.1 Review

With increasing cloud complexity, the amount of effort required for cloud infrastructure management and monitoring must be increased. When compared to traditional infrastructure, the size and scalability of clouds necessitate more complicated monitoring systems that must be scalable, effective, and rapid. In terms of technology, this means that there is a demand for real-time performance reporting while monitoring cloud resources and apps. As a result, cloud monitoring solutions must be sophisticated and tailored to the variety, scalability, and high dynamic nature of cloud settings.

In Chapter 1 we looked at the important monitoring assessment dimensions in depth. Chapter 2 reveals that monitoring solutions in both the open-source and commercial realms do not use all of those characteristics. Each commercial instrument had a specific strength in one of the main attributes listed in 1.1.1. Where particular tools fall short in one property, they usually make up for it in another. We couldn't locate enough information on the monitoring architecture of several commercial tools because they worked on distinct architectures. Monitis and Nimsoft are unique in that they work with both centralized and decentralized architecture, the commercial tools, with the exception of CloudMonix, are able to use a private cloud. Further analysis reveals that CloudWatch and CloudMonix are effective for the vast majority of jobs. When we look at open source tools, we see that many of them are restricted in some form, requiring you to pay a "premium" version of the tool. When a "Free" version is employed, they tend to be lacking in many properties. The tools each have their own strengths, such as collectd, which has excellent performance but no alerting systems.

The state-of-the-art research in the field of cloud monitoring was presented and addressed in this publication. It did so by presenting several design challenges and research factors that may be taken into account when evaluating a cloud computing system. It also discussed a number of cloud monitoring tools, including their benefits and drawbacks. Finally, this work presented a taxonomy of current cloud monitoring technologies,

3.2 Further Research

Because monitoring has become such an important part of the cloud infrastructure, its scalability must be given top emphasis. Based on this fact, as well as the already mentioned monitoring elements and methodologies, we predict that more reliable cloud monitoring systems will necessitate a significant amount of effort. Furthermore, we discovered that there are no universally agreed-upon procedures, formats, or criteria for evaluating cloud monitoring's progress. As a result, we advocate for more

collaborative use of research facilities, where tools, lessons gained, and best practices can be shared with all interested research and professions.

Another interesting part to focus on is the modernization of past research. We discussed that most of the former studies contain tools that are either acquired by other companies or deprecated completely. Thus, comparisons between the tools may be slightly inaccurate. The evolution of the state-of-the-art deserves more recognition and more research. Therefore, we propose modern research on today's state-of-the-art.

Surprisingly, there are no past studies on individual tools. World-renowned tools like CloudWatch and LogicMonitor have not been researched at all. They are only mentioned in review and comparison studies. Furthermore, open-source tools like Nagios and Zabbix are discussed in papers about distributed and cloud monitoring in general, or as part of a review. Studying scientific papers based on individual tools will be great for evaluating the state-of-the-art comparison. We propose further research, separate for each tool, to showcase their uniqueness and their qualities. Finally, each study should provide examples of use.

4 Conclusion

Cloud monitoring is essential to keep up with the huge volume of data that modern cloud systems produce. Many cloud monitoring systems, either commercial or open-source, succeed in their tasks. By taking into consideration past research, we presented those tools and discussed their characteristics and limitations. We evaluated them, with respect to the evaluation dimensions set. Finally, we compared them based on the cloud monitoring taxonomy, the cloud monitoring properties, the monitoring architecture, and the deployment model. All tools discussed show extremely high complexity. There are a plethora of past studies exploring cloud monitoring tools. Despite that, we believe that bibliography needs modernization, because it has not kept up with the pace of the cloud systems' evolution. In conclusion, cloud computing has never stopped evolving since its inception and we have to evolve along with it.

References

[1] Amazon cloudwatch. `https://aws.amazon.com/cloudwatch/`.

[2] Cacti homepage. `https://www.cacti.net/`.

[3] Cloudmonix. `https://cloudmonix.com/`.

[4] collectd features. `https://collectd.org/features.shtml`.

[5] collectd homepage. `https://collectd.org`.

[6] Ganglia monitoring system homepage. `http://ganglia.info/`.

[7] Logicmonitor. `https://www.logicmonitor.com/`.

[8] Nagios core. `https://www.nagios.org/`.

[9] Rackspace. `https://www.rackspace.com/`.

[10] Rrdtool homepage. `https://oss.oetiker.ch/rrdtool/`.

[11] Zabbix. `https://www.zabbix.com/`.

[12] Amit Chaturvedi Aaqib Rashid. Cloud computing characteristics and services: A brief review. *International Journal of Computer Sciences and Engineering*, 7:421–426, 2 2019.

[13] Giuseppe Aceto, Alessio Botta, Walter de Donato, and Antonio Pescape. Cloud monitoring: A survey. *Computer Networks*, 57(9):2093–2115, 2013.

[14] Giuseppe Aceto, Alessio Botta, Walter Donato, and Antonio Pescape. Cloud monitoring: definitions, issues and future directions. pages 63–67, 11 2012.

[15] Khalid Alhamazani, R. Ranjan, Karan Mitra, Fethi Rabhi, Prem Prakash Jayaraman, Samee Khan, Adnene Guabtni, and Vasudha Bhatnagar. An overview of the commercial cloud monitoring tools: Research dimensions, design issues, and state-of-the-art. *Computing*, 97, 04 2014.

[16] Mahantesh Birje and Chetan Bulla. *Commercial and Open Source Cloud Monitoring Tools: A Review*, pages 480–490. 01 2020.

[17] Chetan Bulla and Mahantesh Birje. Cloud monitoring system: Basics, phases and challenges. 12 2018.

[18] Ahmed Mohammed Fahad, Abdulghani Ali Ahmed, and Mohd Nizam Mohmad Kahar. The importance of monitoring cloud computing: An intensive review. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 2858–2863, 2017.

[19] Kaniz Fatema, Vincent C. Emeakaroha, Philip D. Healy, John P. Morrison, and Theo Lynn. A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, 74(10):2918–2933, 2014.

[20] Christopher B. Hauser and Stefan Wesner. Reviewing cloud monitoring: Towards cloud resource profiling. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 678–685, 2018.

[21] Chavee Issariyapat, Panita Pongpaibool, Sophon Mongkolluksame, and Koonlachat Meesublak. Using nagios as a groundwork for developing a better network monitoring system. In *2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies*, pages 2771–2777, 2012.

[22] Gregory Katsaros, Roland Kübert, and Georgina Gallizo. Building a service-oriented monitoring framework with rest and nagios. In *2011 IEEE International Conference on Services Computing*, pages 426–431, 2011.

[23] Praveen Kumar, Priyavrat Singh, Sarthak Chopra, Jagmeet Singh Sarna, and Krishna Rawat. Inspection of cloud computing monitoring tools. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pages 361–365, 2017.

[24] Peter Mell and Timothy Grance. The nist definition of cloud computing, 2011-09-28 2011.

[25] Sophon Mongkolluksamee, Panita Pongpaibool, and Chavee Issariyapat. Strengths and limitations of nagios as a network monitoring solution. 05 2022.

[26] Jonathan Spring. Monitoring cloud computing by layer, part 1. *IEEE Security Privacy*, 9(2):66–68, 2011.

[27] Jonathan Spring. Monitoring cloud computing by layer, part 2. *IEEE Security Privacy*, 9(3):52–55, 2011.

[28] Chellammal Surianarayanan and Pethuru Raj Chelliah. *Cloud Monitoring*, pages 241–254. Springer International Publishing, Cham, 2019.

[29] Hassan Jamil Syed, Abdullah Gani, Raja Wasim Ahmad, Muhammad Khurram Khan, and Abdelmuttlib Ibrahim Abdalla Ahmed. Cloud monitoring: A review, taxonomy, and open research issues. *Journal of Network and Computer Applications*, 98:11–26, 2017.

[30] Jonathan Stuart Ward and Adam Barker. Observing the clouds: a survey and taxonomy of cloud monitoring, 2014.

Getting grip on technical complex systems

Bjorn Kouw, Qiyang Zhong, and Tim Schouten
Department of Computer Science
University of Amsterdam
Amsterdam, 1098 XH
bkouw@hotmail.com, qiyang.zhong@student.uva.nl, and tim.m.schouten@gmail.com

Abstract

Re-designing complex systems is a hard task, especially for designers who have little understanding of the technical functionalities of theses systems. By decomposing and abstracting these systems, and applying frameworks to visualize the subsystems, a more clear overview of the technicalities can be obtained. After the necessary information is gathered, the designer can use design patterns to create a new UX-design which is both technically sufficient and has a high usability. In this paper, ten guidelines are presented which aid the designer through this process and help the re-designing process of technical complex systems.

1 Introduction and motivation

The market in cloud computing is growing. In the first half of 2019, the total revenue of cloud computing was $150 billion, which was a 24% increase from the previous half-year. In 2021, 92% of all organizations were at least somewhat in the cloud.Gartner (2019) These figures show that more and more cloud software is being developed and used by an increased audience of organisations. However, one of the main issues current cloud systems and especially cloud platforms suffers from, is that this software frequently lacks usability Brian Stanton et al. (2015)Adam Belloum ([n.d.]). A cloud platform can be defined as a third-party provider that is responsible for delivering hardware and software tools over the internet, where cloud platforms are necessary for development of application Dou et al. (2013). The service provider hosts its hardware and software on its own infrastructure. The result is that the client's developers only have to install hardware and software to develop a new application. Even though this level of service provision is useful, in various cases the provided interfaces are confusing due to the level of technicality present in the tool designs. When systems lack usability, it will take longer for the developers that work with this software to complete a task Jakob Nielsen (1995). This could lead to increased development costs for the client, or the client will choose to work with another more usable cloud platform.

The people responsible for the usability of the cloud systems are the human-computer interaction (HCI) practitioners which are also known as UX designers/researchers Alexander Jones and Dr Volker Thoma ([n.d.]). A UX designer should have a wide range of skills, from understanding human psychology to requirements modeling and wireframing. UX focuses on issues such as ease of use, ease of learning, user performance, gathering requirements, and user satisfaction. The UX practitioners work together with Software engineers. These engineers are trained in programming, data structures, and database design and are therefore responsible for implementing the designs from the UX professional to running code Thomas Memmel et al. (2007). Unlike software engineers, UX practitioners on the design side of the field in most cases do not have the sufficient technical knowledge to work with the cloud since many design education programs focus on subjects such as the distinction between designing for mobile and desktop applications and the techniques from social science. This lack of understanding is a problem since designers need some familiarity and expertise to function and contribute to the workYang et al. (2018). Cloud platforms are technically

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

and therefore functionally complex. Due to a lack of technical knowledge, a UX practitioner can get easily overwhelmed. This then could lead to a lower quality design and User Interface (UX) Qian Yang (2018) Yang et al. (2018). A use case where UX design provided value in a cloud project is Adobe Creative Cloud. Adobe took all its software and sells it currently through an licensing model in the cloud. Before cloud, these kind of subscriptions were not possible. Then, UX designers needed to have an good understanding what the boundaries of cloud are in order to come up and design functionalities that add value to the user journey Yang et al. (2018).

This paper aims to increase the understanding of UI and usability design in systems where someone with little or no technical knowledge would be overwhelmed when a complex system, such as a cloud system, has to be re-designed. In order to do this, a literature review and case description are done to explain how certain guidelines can be applied to complex system designs in order to increase usability and understanding of the software. The guidelines suggest some working methods that will help the designer deal more effectively with the technical complexity of these types of systems, aiming to help the designer create higher quality designs. These guidelines won't replace the methodologies and rules that come with the design process but add more to the knowledge a professional designer already has. In order to scope this research, the following research question has been defined:

- How can a UX designer better understand the technical aspects of complex software systems when redesigning these types of systems, in order to provide more effective designs with a higher level of user experience?

For a more in-depth insight into the problem scope, a selection of sub-questions were also defined:

- How can a UX designer get better understanding of technical complex systems?
- What should the collaboration between different UX designers and developers look like to create understanding of the system's technical complexity?
- What are UX design patterns to give the UX Designer handles to work with and simplify the technical complexity for the user?
- What are guidelines that UX Designers can adopt in their work process to deal with technical software complexity?

This paper contains four chapters. The first chapter defines complex systems through the introductory explanation of cloud-based systems. The second chapter then explains how Complex Systems can be understood through the eyes of a UX designer. Different techniques of understanding these types of systems are introduced along with a framework which combines the techniques into a way-of-working. Next, this chapter discusses how UX designers can actually obtain technical understanding of the complex system through communication and collaboration within the development team. The second chapter concludes with design principles designers can abide by when actually designing the system after all necessary understanding and information is obtained. In the third chapter design principles on how to work with complex systems are presented. The fourth and final chapter of this paper provides design guidelines which can be used by designers when redesigning technical complex systems based on the information gathered in the previous three chapters.

2 A Cloud System as a Complex System

The definition of a complex system is a collection of diverse adaptive elements that interact with each other, whose micro-level behavior produces macro-level behaviors Chaffee and McNeill (2007). Examples of complex systems are the human body, the economic system, ecosystems, Artificial Intelligence, and Cloud Systems Cindy E. Hmelo et al. (2000). A cloud system is defined in this paper as a complex system. Cloud and especially cloud platforms are complex because cloud platforms allow users to have a high level of control in a dynamic world with lots of variables that interact. A cloud platform is software that refers to the operating system and hardware of an internet-based data center server. It allows letting software and hardware products to co-exist remotely and at scale. A user can, for example, change variables like storage, networking, and CPU in order to get the right balance between costs and performance. Also, the cloud consists of separated services that did not exist before the cloud needed to work together in the system. Other vendors can also offer some services. All these variables that influence each other need to be managed in a cloud platform. That's

why this software is complex Adrian Bridgwater (2021) Chunye Gong et al. (2022). Furthermore, the cloud is by itself a complex technology. By looking at the characteristics of cloud, it can be seen that many cloud applications run on multiple virtual machinesIsha Upadhyay (2020). These virtual machines are, in this example, the elements of the system that need to interact in order to make the application work since they carry each part of the application. There are also backup virtual machines for the situation that one virtual machine will be done, then the copy will be activated. Using virtual machines makes the cloud scalable, which means that new services and copies can be added easily. In other words; the elements in the system are adaptive M. Hasan Jamal et al. ([n.d.]). Since cloud platforms and other cloud systems can be defined as complex systems, the authors of this paper will generalize this research since it should not make a difference in the design process if an UX designer will re-design a cloud system or another system that is complex in a technical way.

3 How to understand complex systems

Complex systems are difficult to understand since they consist of multiple components that interact, coexist and requires management. Furthermore, these systems are often dynamic and interdependent. For people with little technical knowledge, getting a grasp of these systems and understanding exactly how they work takes a lot of cognitive effort and is difficult since some interactions are invisible and/or have a time sequence that makes the processes hard to perceive Cindy E. Hmelo-Silver et al. (2007). When UX designers are required to design such systems, it is important for them to understand how these systems operate in detail. This enables the designer to create designs that have the highest usability possible for users who work with these technical complex interfaces and systems Yang et al. (2018). This paragraph discusses different methods designers and development teams can apply in order to enable the designer to create these high-level designs.

3.1 Decomposition and abstraction

As the name suggests, complex systems contain a lot of different interactions, information flows, and other complicated processes. The totality of these processes leads to a system being hard to understand in its totality. To enable a designer to better understand a system and the different processes, the system should be divided into various different sub-systems Nicholas R. Jennings (2001) Cindy E. Hmelo-Silver et al. (2007). This decomposition of the total system into smaller parts, with each interaction, flow, and process documented separately, makes it easier to understand the system as a whole. The complex system will be more manageable since the designer can treat the sub-systems in relative isolation. In the case of UX design, this decomposition of the larger system into smaller parts enables the designer to focus more on the interactions the user has with the system instead of the content of the system. This makes for a more usable design and also enables the designer to connect the different sub-systems together more easily and effectively. The decomposition of the system can be aided by another method called Abstraction. Abstraction is the method of creating a simplified model of the system which enhances certain properties while it (temporarily) ignores other, less important, properties in the system. By creating a simplified model of the system, decomposing becomes easier because the key elements of the system are identifiedNicholas R. Jennings (2001). The combination of these two creates a more clear view of the (infra-)structure of the system and enables the designer to focus on the usability and functionality of the system together, instead of sacrificing one for the other due to a lack of insights. This means that parts of the system that are not important to know for the UX designer must be left out of the system, as a general guideline everything that won't influence the functionality boundaries of the system can be left out. Applying these two techniques however also has a downside. In some cases, the system is oversimplified, leaving out key functionalities and features which are required for proper usage of the system Cindy E. Hmelo et al. (2000). Instead of a high-level usable design, a feature-lacking design is created which inhibits the proper functionalities of the system.

3.2 Frameworks for understanding complex systems

In order to make sense of complex systems, frameworks have been created to aid developers and designers in creating these types of systems. One example of this is the Function-Behavior-Structure-framework (FBS). "Function" refers to the purpose and functionalities of the (sub-)system, for example, the log-in system of a software application Cindy E. Hmelo et al. (2000). In software

engineering, the “Behavior” then can be seen as the language of causality, where some actions cause some states enabling another action, possibly in a different subsystem. Finally, “Structure” refers to the physical structures of the system and is the part which novices understand the easiest Cindy E. Hmelo et al. (2000). In software development, this could for example be a database and the data it contains as a physical structure, or various front-end components. The “Behavior” is the most difficult part to understand for people with a lower technical understanding of systems because this requires an understanding of both the “Function” and “Structure” parts of the system. The FBS framework is promising because it focuses on the casual understanding of the relationships among different aspects of the system. This is how experts of a system view complex systems as well, where technical novices focus on visible elements and simple causal understanding Cindy E. Hmelo-Silver et al. (2007) Cindy E. Hmelo et al. (2000). Breaking down a complex system to a structural, behavioral, and functional level will help designers understand the complex system without oversimplifying it. Developers can use the FBS framework as a way of explaining systems in presentations, chats or visualizations Cindy E. Hmelo-Silver et al. (2007). In this chapter, the focus will be on the FBS as a system visualization aid which serves as a single source of truth in order to get a shared understanding between the disciplines and to get an overview of the whole system.

Earlier research suggests to only use the functions of the system in the visualization. The authors of this research would like to propose to make use of the FBS technique, since this technique helps to let novice people have a better understanding of the system and give them the tools to gain more expert insights and understanding Thomas Memmel et al. (2007). However, visualizing software according to the FBS framework could lead to many small functionalities that interact with each other, which could lead to confusion due to the level of complexity. This is also known as "Death Star" architecture Kevin Sookocheff (2022). When a system has a lot of functions and interactions, grouping sub-systems together and treating and documenting these units at a higher level could aid in creating structure and understanding of the system. After this, these systems could be broken down again and treated as their own once more. This creates a structured approach to understanding technical complex systems and the relations and interactions between the various levels of sub-systems. In business this is done by decomposing teams into different responsibilities. For example; in an financial institution team one would be responsible for the design and development of the app and team two would be responsible for the website and its functionalities. Another way to prevent the Death Star to occur is to simplify the system visualization in such a way that the person who looks at it still gets the information necessary for the job to be done **?**. In the example of the financial institution the app team needs a more detailed overview of the app subsystem then the website subsystem. In figure 1 an example of an FBS visualization in high overview can be seen. The categories in this figure are separated and form their own sub-system lines, which show how these systems are related. For a real project the contents in the categories behavior, structure and functions should be described in more detail than in the figure in order to get a meaningful overview of the system.

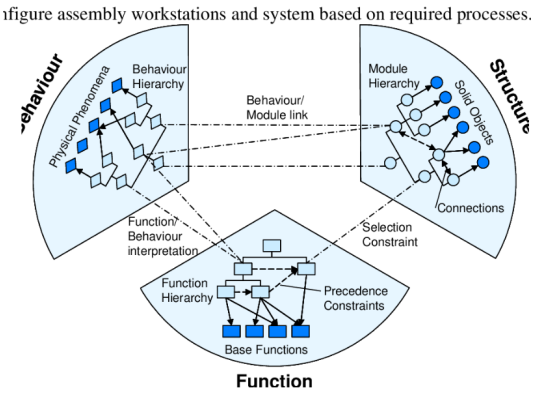


Figure 1: Example of FBS visualisation Sanderson et al. (2019)

3.3 Stock and flow visualizations with the FBS framework

Another way of viewing technical complex systems are as input-output mechanisms. Software systems can be considered as open systems. This means that the system accepts input from outside.

A typical input for software systems would be user input. Inside the system entities and relationships form an unified whole and can be seen as the boundaries from the surrounding world. One can visualize this as can be seen in figure 3. This way of simplified thinking and visualizing input and output comes from the scientific field of system dynamics and is known as the Stock flow Diagram, which is used to map complex systems Thomas Binder et al. ([n.d.]). In this image the input is visualized by an arrow and description which is the input pointing to a dotted box. This represents the inner system with its entities. The arrow pointing the opposite direction is the output Kevin Sookocheff (2022). This is a simplified overview compared to the FBS diagram and is composed of a single interaction. One downside of this diagram is that it can be oversimplified.

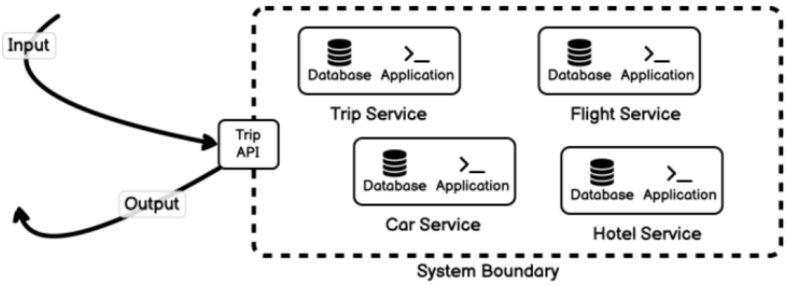


Figure 2: Stock and flow diagram for software systems Kevin Sookocheff (2022)

As discussed earlier, oversimplified overviews have multiple disadvantages Cindy E. Hmelo et al. (2000). One can make this diagram more specific by making annotations about the entities. Another way would be to use the lessons that are learned from the FBS framework. Since the entities are the structure (S) in the FBS framework one could add arrows with descriptions that describe the behavioral relationship between the structures and the outcome (B). To implement the function (F) in this visualization one could add a user story. This agile method describes the problem and the software feature(s) from a user perspective, shows the value of the feature(s) and allows the connection of multiple interactions since the user story can be divided and into several interaction moments. Each moment will have its own stock and flow visualization because the stock and flow visualization can only exist of one interaction. Using user stories tackles the problem when the system consists of a lot of interactions Thomas Memmel et al. (2007). User stories also help to manage complexity since they focus on the user's tasks rather than the content which in some use cases can be difficult to understand for one without specialized knowledge of the system Davide Bolchini and John Mylopoulos (2003). An advantage of this visualization technique is that it helps to compose the system into meaningful chunks which are less complex compared to the whole FBS system visualization that is mentioned earlier. Another advantage of this is that it builds on the existing agile methodology of user stories which are well known by designers and developers Thomas Memmel et al. (2007). A disadvantage is that this method of visualizing the system only allows for visualisation of the sections that are part of the user story, and it cannot show the whole system at once. The visualization method mentioned in the abstraction section is more suited for this. It is advised to make a visualisation according to the method which is described in the previous section to create an overview and shared understanding of the system for everybody in the team and to use the visualization method described in this section to create a deep understanding about an specific part system when the UX designer needs to work with it on project basis Cindy E. Hmelo et al. (2000).

In this paper the stock and flow visualisations together with annotations based on the FBS framework are proposed as a way of visualizing complex software systems. A use case is presented to make the FBS stock and flow visualization from the previous section more tangible. A user story can have various levels, where high-level user stories consist primarily of multiple interactions, and low-level user stories consist of mainly one interaction per user story. This paper uses an example of a high-level user story where somebody wants to change his account setting but first needs to log in. In figure 3, the stock and flow diagram is implemented with a use case. The use case is simple, but provides an example which describes that this way of visualizing systems together with use scenarios and explaining the behavior can reduce the complexity of technical systems. The texts explain the relationships according to the FBS framework. However they can also explain necessary domain

knowledge since this can be necessary to understand the elements from the FBS framework Cindy E. Hmelo-Silver and Roger Azevedo (2006).

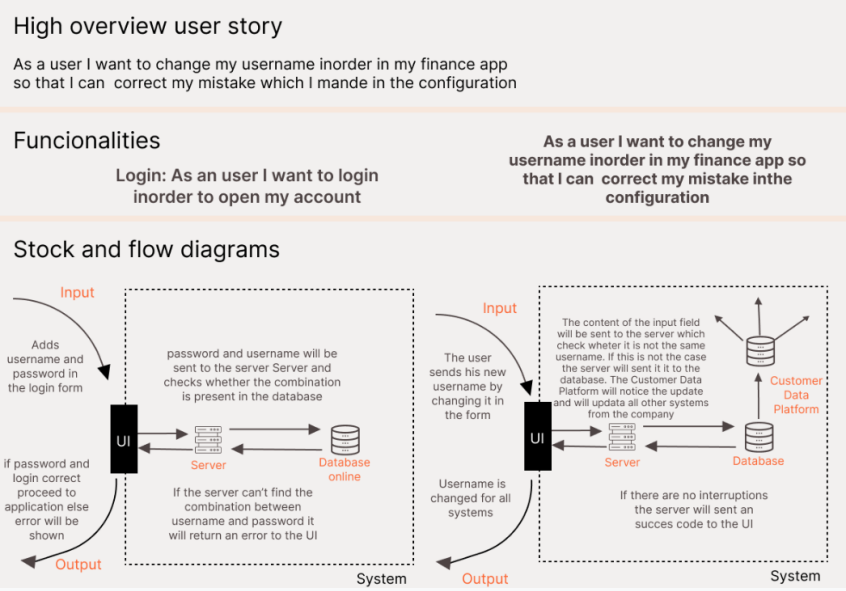


Figure 3: A stock and flow use case

3.3.1 Receiving data from another system

Sometimes a system needs to receive data from another system to work. From figure 4, it can be seen how this can be accomplished. Since this is not a use case, the elements from the FBS framework are not displayed. Defining when structures need to be displayed as a separate system is defined by the system and its boundaries, and the best solution differs per use case Richardson and Lissack (2001). This way of visualizing two interactions is common in system dynamicsBernhard J. Angerhofer and Marios C. Angelides (2000). An example of a possible system boundary is where the team's responsibility ends. This separated system can have another level of abstraction, but this also depends on what the UX designer needs to understand the system they are working on. When the other system is not necessarily required to be understood, such as an external microservice, then it could be treated as a black box that sends data.

4 Communication across disciplines

Understanding an existing system is the first step in re-designing complex systems. In order to understand the requirements for the redesign, information regarding the new technical infrastructure has to be acquired by the designer in order to create a design that serves these requirements. This transaction of information is key in the process of redesigning complex systems because the technical infrastructure determines the various sub-systems and interactions present in the total system, and therefore determines the final design. The key factor that determines the success of the information transaction is good communication. Research has shown that among project teams with different backgrounds and tasked with complex system redesign, good interdisciplinary collaboration is crucial to the success of the project by "reducing inefficiencies and avoiding sub-optimal products being released" . In order to achieve a successful level of communication, close cooperation and proximity within project teams are essential Alexander Jones and Dr Volker Thoma ([n.d.]). The source of the challenge in interdisciplinary collaboration on technical complex systems is primarily from collaborators' different levels of background knowledge of the products. For example, computer engineers could build complex software systems but would need (UX) designers to create designs to make the product presentable, marketable and make sure the product is easy to use for customers. To design such a showcase would require designers to have knowledge of the technical aspects of the software system and engineers to have knowledge of the notion of usability and general design systems

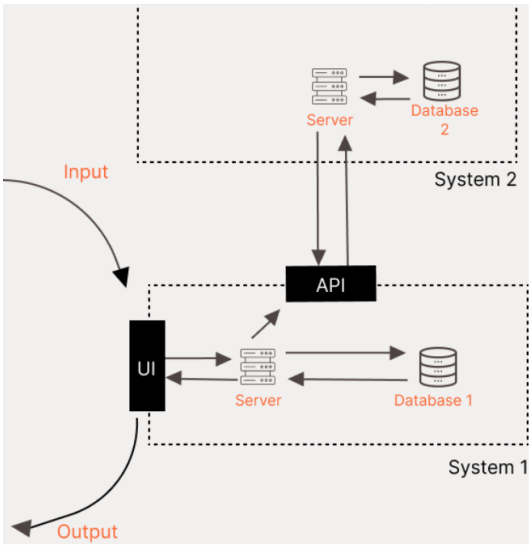


Figure 4: Receive data from another system

Alomari et al. (2020). However, it is shown that teaching STEM technical material to non-STEM students is difficult even in the classroom and would require novel teaching methods to have some success MAJ and NUANGJAMNONG (2020). In the interdisciplinary workspace, then, it would be challenging for this kind of background knowledge to be acquired by the collaborators in a clever, and cost-efficient fashion. With the obstacle of background knowledge, efficient communication, workflow design, and workload distribution, is ever more important in interdisciplinary collaboration to communicate effectively. It can be shown that none of these general requirements can be missed, and one requirement facilitates the other in improving the development process with the advantages they bring to the table. In the headers below, the importance of the different aspects of interdisciplinary communications are explained and discussed.

4.1 In-person Communication

Various studies delve into measuring the efficacy of collaboration among different disciplines In health care, a successful collaboration between nurses and medical engineers would involves a brainstorming session where all of the coworkers would contribute to a new detailed design Zhou et al. (2021). The designs are then "revised until all team members have no further comments". Postdoctoral researchers from Columbia University Winowiecki et al. (2011) with diverse backgrounds working on a research question would employ several strategies to facilitate their collaboration, which includes communicating during in-person seminars and engaging in fruitful and deep epistemological and philosophical discussions. Without these seminars, it is hard to imagine how they could engage in deep dialogues. In person communication, then, would prove to be beneficial or even mandatory in facilitating interdisciplinary collaboration in these kinds of situations. A paper Thomas Memmel et al. (2007) discussing agile developing process for software and UX engineering argues that in the workspace of software engineers and UX designers, "continuous and lively discussion is necessary [...] Informal communication across organizational borders should be easy, and teams should share offices and common spaces." This image of an open workspace is different from a traditional classroom, where in traditional teaching the materials would have a high Intrinsic Cognitive Load and causes materials to be hard to learn and remember MAJ and NUANGJAMNONG (2020). Knowledge is relational when it builds open upon other pieces of knowledge, such as in the situation when someone learns about how firewalls prevent attacks would also need to learn about OSI transport layers and a UDP/TCP protocol for example MAJ and NUANGJAMNONG (2020). High Intrinsic Cognitive Load is the result of teaching highly relational knowledge, which happens to be the case for a lot of the background knowledge required to understand cloud systems, in limited teaching time and with limited support from teachers MAJ and NUANGJAMNONG (2020). In the open workspace, knowledge and ideas could be circulated without the traditional teaching pedagogy and argued as more easy to grasp for teammates with these other backgrounds. The freedom of

engaging in personal, face-to-face dialogues means the dialogue participants could organize their ideas and materials freely and provide better learning support for the other participant, which results in lower Intrinsic Cognitive Load. Whereas teachers teaching a large classroom would be forced to use traditional teaching methods. Furthermore, in the context of software and UX designers collaboration, including designers into the development process and the other way around is crucial and it is facilitated by the open workspace. The main goal of the design phase is that the designs that are delivered will provide a good user experience, match the requirements of the stakeholders and be able to be implemented by developers in a technically feasible manner. Previous research showed that designers produce higher quality designs when they work together with development when working with complex systems, since development can help them to understand the boundaries of what is possible with the technology, it is important that these communication happens face to face Yang et al. (2018).

4.2 Division of Roles

Open workspace is not advocating for unorganized anarchy. Clear roles and capable leadership exist within agile frameworks as well. A leader "should be an authoritative person who must have a deep understanding of both subjects Thomas Memmel et al. (2007)." Another paper about UX design in agile advocates that the project manager (PM) should have experience of UX design and technical development Laura Plonka et al. (2014). A capable and knowledgeable leadership would not only simply solve the teams' development and design obstacles by "navigating through the development process, proposing solutions to critical design issues and applying the appropriate design, engineering and development methods", but also facilitate their collaboration and in the end improve the final product. For example, the project manager would know the importance of communication in interdisciplinary collaboration and encourage a friendly, open workspace as well as organize activities for colleagues to participate in. The team would know that it is being guided by the right hand and view the project manager as an example that it is possible to be knowledgeable in both fields and have confidence in the learning, communicating and developing process. On the management scale, a "Project Manager with a background in technical and UX management mean[s] that the project level decision maker [is] sensitive to the team's UX challenges and [is] able to introduce new ways of working to improve the situation"Laura Plonka et al. (2014). Sensitivity is another quality that is useful for the leadership in interdisciplinary collaboration. With a timely intervention, it is possible to avoid a delay of the teams‘ deliverables and a would-be development disaster altogether. Sensitivity would not simply be a standalone quality of the leadership, it also requires good communication for leadership to have the capacity to be sensitive.

However, it is possible in an open workspace that a person does not take advantage of the workspace and prefers to isolate themselves with people who share the same background. Therefore, various communication problems could still exist. An example of this comes from earlier research, where "developers [were feeling] inhibited about phoning a designer to discuss [an] issue" and preferred indirect communication, "or tried to resolve the issue within the development team for the sake of speed". The designers, not knowing the issues, would be unlikely to discuss and discover this issue even in open space due to this lack of communication. This type of problem could be solved with various methods such as organized meetings. Another method is assigning a Business Analyst (BA) who could serve as a communication bridge between clients and the interdisciplinary teams Laura Plonka et al. (2014). "This BA role [could be] staffed by a senior developer, able to manage the discovery and communication of requirements and to provide direct feedback on the technical feasibility of design ideas coming out of the meetings between designers and clients." With a senior staff members handling these meetings, the outcome of these meetings would be fruitful. This would save more time compared to letting the teams talk to clients themselves. Feedback from the BA would be valuable for the teams' internal communication. The team would consequently know what to communicate. Their communication would aim to improve upon the BA's feedback, which further facilitated their work process.

4.3 Organized activities

Various researches discuss agile as the framework for interdisciplinary collaborations, especially among computer engineers and UX designers. Agile, however, is only a framework with different practices. The detail on how to implement agile affects the performance of interdisciplinary projects Laura Plonka et al. (2014). Nevertheless, it could be argued that learning tasks that facilitate learning

the background knowledge are mandatory to implement in interdisciplinary collaborations within agile frameworks. Earlier research did an experiment where middle school students had to design an artificial lung, but they first had to understand how the complex system of the lung works. This experiment was based on design based learning that practitioners learn about a given problem and design a solution for it. It is important that the assignments has some modeling aspects, to make an artificial lung means someone has to have an understanding on how the lung interacts with other organs for example, which will be replicated in the end product. The difference with the traditional way of classroom learning is that in design based learning students are constructing knowledge instead of passively receiving knowledge, this is beneficial since it is more engaging for them Cindy E. Hmelo et al. (2000). Although this argument might not be relevant for UX Designers since their motivation is different than middle school students, the writers of this research expect that using a familiar methodology to conduct knowledge will make it easier for them to understand a complex system. Making an artifact during the process will also make the knowledge seem less abstract than only teaching the theories of the system. The results of the experiment showed that students following a design based learning approach understood the complex system of the lungs better than the students that followed the traditional teaching approach. The researchers also state that reflecting with an expert of the complex system about what is learned in the design based learning process is important since this tackles misunderstandings and the expert can give feedback on the learnings Cindy E. Hmelo et al. (2000).

Earlier research showed that feedback and iterations are an important aspect of the learning process Janet L Kolodner (2002). Therefore, at the start of the project tasks will be set up to facilitate new UX designers' learning in addition to open workspace. An example of such a task is letting the designer takes charge of a system and creates a quick re-design. The assignment must be a practice assignment on a real project to ensure that learning the system is the essential part of the actual re-design. The purpose needs to be communicated to the UX designer to understand that less time needs to be spent on the outcome. Someone who is an expert in this system needs to guide the designer. This could be the project manager or business analyst. The UX designer will give presentations to the mentor, and the mentor will give feedback if he understands the system correctly. The UX designer will make iterations on the design based on the developer's feedback. The time the UX designer gets to complete the practice assignment is dependent on how big and complex the system he will re-design, but time needs to be scarce enough to let the deliverable be of secondary value but to make iterations in the process possible.

Other types of activities that promote group communication are recommended. Examples of such activities are daily stand-ups or design studios. In daily stand-ups, teams and the person in charge, which is usually a product manager, will attend the meetings and report their work progress. It provides a venue of communication and keeping up to date with each others' work Laura Plonka et al. (2014), which should prevent the issue of designers or developers isolating themselves and preferring solving issues within their own teams. Therefore, it is important that both designers and developers join the meeting, and it is also important that persons working on the actual product attend the meetings consistently. This can prevent the phenomena that designers attend meetings but them often not being the designers currently doing the work Laura Plonka et al. (2014), which would cause the interdisciplinary team and the leadership to lose track of the progress on designers' work. Design studios are meetings where designers, developers and stakeholders produce design sketches, present them and critique them in order to find the best technically feasible solution with the aim of promoting communication, shared understanding and shared ownership of designs Laura Plonka et al. (2014). With a shared understanding, the aforementioned knowledge and communication gap between the interdisciplinary teams could be reduced. The team members would essentially be doing the stated learning tasks that practice their designs on prototyping and construction of new knowledge. With regular design studio meetings, they would also get feedback and iterations on their own work, which facilitates their learning process. With a promoted shared ownership, they would be increasingly encouraged to take responsibility for their own work and consequently they would be less likely to take actions that are convenient for them but damage the product as a whole. For example, preferring indirect communication instead of participating in face-to-face conversations is an action that is convenient but damaging, and it shows that teams taking such a action do not think about the complex product in its entirety and only want to finish their tasks on their subsystems.

5 Design principles

Where the previous chapters of this paper were focused on the process of understanding a technical complex system, this section describes the process of designing for complex systems. The principle of interaction design is about realizing a human-computer interface of which the usability is positively perceived by the user. In this case the interface is a Graphical User Interface (GUI). The most important characteristic of a GUI is the direct manipulation. This enables users to interact with the elements on the screen by touch or a pointer device. However most GUIs are a mix between direct and indirect manipulation, for example by clicking on the menu items the menu opens, which is the direct manipulation, from where a menu item in selected, the indirect manipulation. The GUI has some advantages(for example the predictability of the system, reversibility of actions and task completion time), however the effect of these advantages are dependent on the quality of the UX designer's work. This chapter will help the UX designer make decisions to realize these benefits when they are working on a design of a complex service. In this section a multitude of design principles for complex systems will be introduced and discussed.

5.1 Principle 1: Use patterns from web design wherever applicable

According to Alan Cooper, one of the founding fathers of usability, the difference between interaction design patterns from a simple conventional website and a complex system should be as small as possible, and the design approach should not differ as well M.T. Hoogvliet (2008). An example of this is a simple radio (option) button. If a user has three to five options to choose from, and only one can be selected he, the radio button is a suitable interaction design pattern, as this forcefully limits the user to only a single choice. This means that in the design of a technical complex system that is software based the interaction design patterns that are also known from the web should be used, since users transfer their knowledge from the web to this complex software system. Examples of web design principles are design a clear application, let the application forgive users their faults, do user research, test the design and create a consistent design. There are more general design principles but since this paper is focused on creating guidelines for working with technical complex systems it is assumed that the general principles are already familiar Luke Wroblewski (2001).

5.2 Principle 2: Identify the Posture

It is good practice to understand what kind of system will be (re-)designed since this helps to choose the design patterns that are the most appropriate for that specific situation. Research found that graphical user interface applications can be divided into four different types of systems:

- *Sovereign postures*: Sovereign postures are screens that are mostly used in full screen or enlarged mode by users. This type of interface is suited for interfaces that require a lot of interaction for prolonged periods of time. Typically this type of software is a tool that users use to do productive tasks. An example of a Sovereign posture is a text editor like Google Docs.
- *Transient postures*.:Transient posture interfaces usually serves as a support supplement of a Sovereign posture. This type of posture is mostly used shortly by its users to perform a single task and then return to the sovereign posture. Because users do not use these transient postures as much as sovereign postures, controls should be more stressed and bigger than sovereign postures. An example of an transient posture is the calculator on MacOS.
- *Auxiliary posture*: The Auxiliary posture is a combination between the sovereign posture and the transient posture. They are always visible but only play a role of support. The task-bar is a good example. Often these programs are quiet process reporters which notify the user in a simple process that is currently going on. It is important that these messages respect the sovereign posture.
- *Daemonic postures*: These postures are program interfaces which rarely perform interaction with users. Daemonic postures perform tasks behind the scenes without users being aware of this. If users need to interact with such a system they are often perceived as difficult to use. Printer drivers are an example of a Daemonic posture.

Once the type or types of appropriate postures are identified, the designer can change the design patterns which match the appropriate posture. An example is default window size or, if the functionality

which he is designing is a support element, he can choose to do it in a pop-up instead of the main screen Marton Sakal (2009).

5.3 Principle 3: Make clear to the user a specialized environment is used

According to previous research, hiding the browser controls makes it more obvious for the user that he is working in a specialized environment rather than just a standard website since it has the look and feel of a desktop application. The authors argue that the maximum information communication capabilities will increase by removing the browser functionalities. However, the authors of the current paper question whether the research findings from 2001 are still valid. The assumption is made that users are now more experienced with using complex tools in the browser than in 2001 and therefore understand that the browser can do more than only browsing standard websites Luke Wroblewski (2001).

5.4 Principle 4: Minimize the use of browser windows

The user should be able to use multiple browser windows. This is by default possible by using HTML for example, where the user is enabled to perform multiple tasks simultaneously. However, when the system opens new browser windows by itself, there is a risk that some browser windows will get lost behind others, leading to confusion and disorientation. The user should take action by using the browser's standard functions whenever a new window is required to open Luke Wroblewski (2001).

5.5 Principle 5: Use Rollovers

Information overload will happen when too many interface elements are shown on the same page. Rollovers are used on the web to eliminate unnecessary interface texts that only serve for more information or clarification that more experienced users might not need. These rollovers can be activated by hovering or touch on mobile or tablet. However, using too many rollovers will result in a flickering effect by constantly having items appear and disappear. It is also not a pleasant experience for the user to use a rollover for every interface element in the system to understand what is going on, so use them only when necessary. When interface elements can be explained in one short sentence, an ALT rollover might be a better option, but this functionality is not available on mobile Luke Wroblewski (2001).

5.6 Principle 6: Task-based design

Complex software systems are in many cases used to fulfill a task. Examples are applications from financial institutions, cloud platforms, and text editors. The UX designer should work task orientated, due to the many different ways an application can be used. The user's goal should be defined and this goal has to then be decomposed into into smaller goals and functionalities, making the complexity of the system more manageable. Task based design is not often used in web design since communication means are more important in websites than in complex software systems. Luke Wroblewski (2001) Thomas Memmel et al. (2007) Davide Bolchini and John Mylopoulos (2003).

5.7 Principle 7: Aesthetics

Visual design can give a complex software system personality, familiarity, trust, and user enjoyment. However, by using non-functional or changing the website patterns for aesthetic reasons, the usability of the system will suffer. It is important to not allow visual treatments to overwhelm interaction elements and to only use visual design for functional reasons. Using animations in the visual design can also increase usability when this is done the right way. Animation should be used as a feedback mechanism only to show, for example, the transition between screens, illustrate change over time, or attract attention. Poorly executed animation will only distract the user. However, the system should not solely depend on animation for understanding since some users can deactivate the animations using their browser settings Luke Wroblewski (2001).

5.8 Principle 8: Simplification

The design process of a UX designer whose task is to redesign a complex system will do this by using his knowledge based on previous experience. Simplifying the system wherever this is possible will shorten the time needed to understand the system and therefore, as a consequence, reduce costs. It is important to ignore interactions in the system and information that have no influence on the behavior of the system, or do not provide information about the technological boundaries Luke Wroblewski (2001).

5.9 Principle 9: Scaffolding

Scaffolding is support that enables people to accomplish tasks they could not do otherwise. When a scaffold is well designed, it will help people learn and discover new tasks of the system and increase their competencies. It is essential that the scaffolding layer will fade and disappear over time and perhaps sometimes return so the learner can complete a task independently. Scaffolding needs to assist in the use of questioning, prompting, modeling, and discussing. An expert will take this role in a normal situation, but sometimes this is not possible. Technological scaffolding can take this role and offer the same support as experts. Scaffolding is sometimes necessary for complex software systems since in some cases the UX designer, for various reasons (such as the topic of domain not able to tackle all the complexity for novice users), a learning curve might be required. Scaffolding helps the user to overcome this learning curve. Three forms of scaffolding can be identified:

- *Communication and Guidance* This type of scaffolding means that the process of a task will be demonstrated or modeled to the user in a simplified way. Also, the task can be structured into smaller separated tasks by, for example, checklists. This way of scaffolding will guide the user.

- *Coaching* While users perform tasks, guidance will be given by feedback and suggestions from the system. This can be done by highlighting certain parts of the interface for example, giving statements that help to solve the problem. Often users can make use of coaching on request.

- *Eliciting articulation* This type of scaffolding asks the user to reflect on their given task and to show their way of thinking. This can be used as a basis of discussion. An example in software engineering are code comments.

In software, these types of scaffolding can be separated into glass-box scaffolding and black-box scaffolding. In black-box scaffolding, the software performs an action instead of the user, usually because learning this task is not important in order for the user to achieve their learning goals. An example of black-box scaffolding is a calculator in an e-learning environment. Doing the computations by hand might not be the learning goal. However, when the user performs well, the calculator might disappear. Black-box scaffolding simplifies processes without increasing the understanding of the processes. Glass-box scaffolding lets students focus on their learning goals before dealing with other sub-goals and communicates about the processes of the system. Examples of glass-box scaffolding are, for example, prompts and instruction videos that prompt the user to perform the action themselves. Glass-box scaffolding will disappear over time when users are more familiar with the system. Previous research provided a table with examples for each type of scaffolding, as shown in Figure 5.

For the UX designer, it can be advised to identify if scaffolding is needed for the application and if so, which type of scaffolding is necessary. For example, if previous domain knowledge is necessary which the user does not have and cannot be automated, a communication process glass-box scaffolding is necessary for a tutorial. Earlier research found that scaffolding is a promising technique for learning about complex systems when it is combined with modeling exercises since this technique can guide the exploration of the system and can provide necessary background knowledge about the system. However, what requirements these scaffolds should have in a complex system domain is a topic for future research. The scaffolding can be designed to tackle the learning curve of the system for a user of that software, but it can also be used to onboard new team members to make them familiar with the system Cindy E Hmelo-Silver ([n.d.]).

Examples of Glass- and Black-Box Scaffolding

	Glass-Box Scaffolding	Black-Box Scaffolding
Communicating Process	Apple Guide: Explains what to do to accomplish a task and why.	Menu systems: Do not make evident why items are disabled.
Coaching	ThinkerTools Inquiry: Critics that explain rationale or support inquiry processes (White, Shimoda, & Frederikson, 2000).	Wizards in Microsoft Excel: Accomplish tasks under your direction, but do not tell you how to do them without the coaches—the scaffolding is opaque and not meant to fade.
Eliciting articulation	CSILE: Students choose the metacognitive prompts, which they will then use to label their notes—students understand that this is to help them understand the role of their notes in their learning (Scardamalia et al., 1989).	Summary prompts in Microsoft Word: No explanation for why a summary (and other articulations) are being requested.

Figure 5: Scaffolding table Cindy E Hmelo-Silver ([n.d.])

5.10 Principle 10: Cloud Usability guidelines for cloud providers

When the complex system is an cloud platform, the cloud usability guidelines found in previous research should be followed. Not all aspects mentioned in the guidelines will actually make the cloud system more easy to use but it serves as an overview that describes the expectation of the cloud consumer which the UX designer can communicate as user needs to the developers. This research found that cloud usability can be broken down into the following 5 categories and 20 elements. Some items can be considered important for all complex systems, like accessibility, responsiveness and User satisfaction. The categories and elements are:

- *Capable* The following functionalities should be available in the cloud server for cloud consumers:
  - The consumer must be able to see if the cloud service is based on current technology.
  - When the consumer uses an IaaS service, then the consumer must be able to define the exact hardware specifications. For SaaS services the service must be independent of the cloud hardware and operating system in order to work on any software and hardware.
  - The cloud service should be available from every device.
  - The cloud service should offer standard cloud services for cloud consumers like elasticity and scalability.
- *Personal* The application should offer the functionality to change the look and feel and adapt to the needs of the organization.
  - Cloud services should be accessible for people with various needs and characteristics
  - The interface of the cloud systems should be customizable and users have to be able to change the interface(s) to their own need (for example selling an application as an white-label product).
  - The user should be in control of the cloud service, for example being the one that determines if cookies may or may not be placed.
  - Consumers must have ownership of their own data and it should be made clear who has access to the data and how it is used.

  - Multiple access authentications need to be designed to ensure that the user can seamlessly login. It is important that the user shouldn't be aware of all the authentications and authorizations that happen in the back-end to ensure ease of use.

- *Reliable* The system and its functions are required to work under agreed conditions and time constraints.
  - Cloud systems should be highly available. Availability can be defined as being able to perform the functionality over an agreed time period and under certain conditions. Some cloud providers are advertising over 99% availability.
  - The service should be consistent where interaction patterns are applied in the interface, as well as make sure the behavior of the system should remains consistent under different circumstances. It doesn't matter for the user whether or not he uses the application during peak hours for example, but the system should remain as accessible as it is under less traffic.
  - The service policies from the cloud provider should be transparent and easy to understand to the organization of the cloud consumer since this enhances trust. Another way to increase trust is to let the users that work with the cloud system from the cloud consumer team have access to the cloud data center and be informed about the details of the possibilities of cloud platforms

- *Valueable* The cloud consumer expects that the cloud system will add value to the organization and provide various solutions.
  - Organizations expect to save money and other resources when they use a cloud application compared to when they run the service on their own infrastructure.
  - The experience of the users should be good enough so that they keep consuming the service.
  - The cloud service should bring new functionalities that wouldn't be possible in a different IT set-up.

- *Secure* Cloud consumers expect their data and the cloud system itself are secure
  - Cloud systems have to be secure, meaning it must be resistant against various attacks on its hardware and network.
  - Personal data leaks should be prevented. This can be done by authenticating a user's credentials, tools, applications, utilities and patches before providing access to sensitive data.
  - Unauthorized users should not have access to data or be able to start any process. The organisation should be able to decide who has access to which processes inside the system.
  - The cloud provider must ensure that the user has trust in the cloud service. This can be done by listing how authorization, privacy and security are realized in the service level agreement Brian Stanton et al. (2015).

6 Complex system design guidelines

From this paper a summary is made in the form of ten guidelines that can help UX designers to work with and understand complex systems in a more effective way. These guidelines are based on the insights that were obtained from literature research which are described in the sections above.

6.1 Create a visualization of the system and explain it according to the FBS framework

To get an overview of the whole system, it is advised to create an overview according to the FBS framework. This creates a shared image of the system along with the different members of the team. The FBS structure is chosen since this helps readers to understand systems the same way experts do. More information about this guideline can be found in the "How to understand complex systems" section.

6.2 Decompose the system into subsystems to prevent the designer from getting overwhelmed by the complexity of the system

When a system consists of many interacting structures, the user can get overwhelmed by its complexity even when the FBS method is used. This is also known as a Death Star. If this is the case, then it is advised to break the system into various subsystems that make sense. This would mean the reader still gets the system information that is necessary, but in a simplified manner. An example of a meaningful decomposition is when a financial institution, for instance, has an app team and a website team. The app subsystem should be shown in much more detail to the app team. The website subsystem could probably be shown more abstractly or be left out. However, the way the system should be decomposed and if decomposition is necessary would be different for every situation. More information about this guideline can be found in the “How to understand complex systems” section.

6.3 Abstract the system as much as possible and necessary, but no further in order to prevent a too simplified view of the system

Some parts of the system should be described and visualized in more detail than others. Abstractions emphasize parts of the system while it ignores others. This method also prevents the visualization of the system from becoming a Death Star. Making the system more abstract leads to a simplified version of the system. This can be a problem since an oversimplified system can create a wrong understanding of the system or leave out important information which is necessary to work with the system. In the example from guideline two, the app team requires less understanding of the website system than the app system. This means that the app team does not need all the detail of how the website system works. They could visualize the website system as a subsystem and treat it as a black box, where the subsystem is acknowledged to be part of the system but not explained more elaborately. However, how the system is abstracted and if abstraction is necessary will, again, differ per use case and for who is responsible for the redesign. More information about this guideline can be found in the “How to understand complex systems” section.

6.4 Use the FBS Stock and flow diagram to visualize parts of the system during a project to create a better understanding of the subsystem

The visualisation proposed in guideline one is meant to give the reader an overview of the system that is being worked on. However, during a project, a deeper understanding of a specific parts of the subsystems might be necessary. In this paper, a method is proposed that makes use of the user story, which is often used in agile methods to decompose the system into a smaller subsystem(s).The user story will be the function in the FBS framework. From there, a stock and flow diagram needs to be made. These diagrams come from the scientific field of system dynamics and are used to map complex systems. A different stock and flow diagram should be made for every interaction (trigger) in the user story if it differs from the previous one. In the current use case, the system contains all the technical items that are defined in the FBS framework structures. The system arrow pointing to the system is the input once triggered (interaction), and the arrow pointing out of the system is the system’s output. The behavior between the structures when the input occurs is shown by arrows and described with annotations. This proposed way of visualizing the system decomposes the system in a meaningful way. It makes it easier to focus on crucial parts of the system and for the UX designer to understand their tasks. More information about this guideline can be found in the "How to understand complex systems" section.

6.5 Have an open workspace where teams work closely together and share office space, and facilitate in person communication during the collaboration

Knowledge gaps exist in interdisciplinary projects and cause communication problems. To facilitate communication, in-person communication and open workspace is encouraged, instead of passively being taught the knowledge in a mentorship program or in a classroom setting. Engaging in conversations or seminars could help circulate the ideas and close the knowledge gap better since the participants are actively engaging in conversations. In an open workspace it is easy to include designers into the development process and the other way around. Previous research showed that designers produce higher quality designs when they work together with development when working

with complex systems, since the development team can help them to understand the boundaries of what is possible with the technology, it is important that these communication happens face to face.

6.6 Have a product owner with knowledge of all subjects responsible for that specific subsystem and have people responsible for their own roles in the team

Hierarchy and leadership exist in an open workspace. A capable and knowledgeable product manager or owner with knowledge of all subjects would not only simply solve the teams' development and design obstacles, but also facilitate the team's collaboration and in the end improve the final product. For example, the project manager would know the importance of communication in interdisciplinary collaboration and encourage a friendly, open workspace as well as organize activities for colleagues to participate in. The team would know that it is being guided by the right hand and view the project manager as an example that it is possible to be knowledgeable in both fields and have confidence in the learning, communicating and developing process. It also makes sure the developers' views are represented in the designers' team and vice versa. Roles that are responsible for the teams' various aspects of collaboration need to be set up. A business analyst serves as a communication bridge between clients and the interdisciplinary teams. The business analyst works closely with the teams and clients. The business analyst could manage the discovery and communication of requirements, and to provide direct feedback on the technical feasibility of design and development ideas.

6.7 Do modeling exercises as onboarding tasks that help new team members understand the system where they will work with

It has been shown in an experiment that a group of middle school students, whose goal is to learn the inner workings of lungs by designing an artificial lung, learn better than students taught by traditional classroom teaching. It is concluded that in design-based learning students in the first group are constructing knowledge instead of passively receiving knowledge, which makes the learning more engaging for them, which in turn make them comprehend the knowledge better than the second group. Similar tasks could be set up for new team members in the onboarding process. The designer, for example, could be given ownership of a real system and create a quick re-design. Someone who is an expert in this system needs to guide the designer, who could be the aforementioned project manager or business analyst. The UX designer will give presentations to the mentor, and the mentor will give feedback to make sure the designer understands the system correctly. The UX designer will make iterations on the design based on the developer's feedback. Feedback and iterations are important in this process, because they make for an engaging learning experience if the feedback is correct and constructive.

6.8 Have designers and developers join each others daily project stand-ups

In daily stand-ups, all teams and leadership will attend the meetings and report their work progress. This provides a venue of communication and keeping up to date with each others' work and prevent a possible issue that designers or developers isolate themselves and prefer to solve issues within their own teams. In design studios, designers, developers and stakeholders produce design sketches, present them and critique them in order to find the best technically feasible solution. This would promote communication, shared understanding and shared ownership of designs, which leads to improvement on the final product.

6.9 Apply design principles when (re-)designing technically complex systems

Eleven principles have been described that need to be taken into account when the UX designer starts with the design of a complex system. More information about this guideline can be found in the "Design principles" section.

6.10 The UX design process of the complex system should be no different compared to the normal UX Design process

In the usual UX design process, user needs will be researched, a prototype will be made, tested with real users, and iterated according to their feedback and gathered insights. This is also applicable to the design process of creating complex systems. The guidelines formulated in this section do not

replace the tools, principles, processes, and knowledge the UX designer currently uses to design systems. User research, user testing, customer journey mapping, and creating personas are still helpful methods for designing a high-quality application. The reason why these are not treated in this paper is because they are considered basic knowledge.

7 Discussion

7.1 Understanding complex systems

First, ways to better understand technical complex systems have been proposed, along with the FBS framework. This framework is useful for better understanding which aspects of the system should be highlighted and which should be left out. In this paper, it is advised to create two system visualizations during a project: one that contains the whole relevant system, and one where the system is decomposed according to the user story. The writers of this research propose the second visualization, and there is (to the author's knowledge) no scientific literature that proves that this visualization would tackle the complexity. However, this proposed method combines principles where there is scientific proof that it helps to tackle complexity or is used by other scientific principles to help them manage the complexity. Therefore, the authors of this research believe that this way of making visualizations will help the UX designer understand complex software in a better way. These principles are the FBS framework, Stock and Flow diagrams, and abstraction and decomposition. A user story will be used to decompose the system. The advantage of this method is that it is well known by designers and developers and is used in many companies to decompose the work for the UX designer. Adding the system visualization to the user story will not only result in a logical decomposition of the system, but the UX designer can also work with a method he is familiar with. The proposed stock and flow FBS visualization is the first version of how such a visualization based on the user story can look, but one can iterate on this. For example, an improvement point would be to make it less text-based. An assumption that is made in this paper is that there is already a complex system that exists. This means that the complex system and its logic is already present, and the designer needs to redesign the existing system. This is fundamentally different from creating a complex design from scratch, since it would be otherwise hard to map the system. One could ask whether it would be the UX designer's role to understand the system since this can limit creativity. When the UX designer focuses on what is possible with the current components of the system, he might not propose functionalities where the system needs to be changed but are despite valuable input. The authors of this research assume that UX designers will keep doing this when the developers and product owner communicate clearly to the UX designer that this input is appreciated and plan some time for them to design them and give them when necessary input in for example a co-creation session.

7.2 Guidance and communication

After this, the next section described the importance and different aspects of interdisciplinary communication. Here it became evident that in-person communication was very important for information exchange, and a clear division and capable leadership are required for more effective work and encourage team communication. These aspects are positively aided by the so-called open workspace. Another way of stimulating effective teamwork is by organizing activities that not one but all the interdisciplinary teams attend. In these activities, team members learn more about the case they are working on, and obtain more information regarding project progress. Together they develop sketches, prototypes or design ideas with stakeholders. These ideas will then be reviewed by senior members of the team and provide a goal or a standard for the next stage of work and communication. Learning tasks are setup as on-boarding process for junior members to close their knowledge gaps with the rest of the team. These tasks will give them hand-on practice on the real project and would stimulate their learning. Next up, the paper presented design principles which designers can follow in order to create designs with a higher level of usability. These patterns touch upon several different aspects of the design process and factors that need to be taken into account when creating the UX-design.

7.3 Complex systems design principles

These principles are the ones the authors found to be the most prominently represented in literature and good examples of principles to abide by. However, as can be understood from the ways these

principles are described, there are a multitude of other principles which could also be used by designers. Take for example principle 4; "Minimize the use of browser windows". Other types of applications might not use browser windows to operate, but other functionalities to which certain principles could be applied. This shows that a lot of case-specific "principles" could be formulated, but this paper tried to highlight the most prominent ones in a generalized way. In principle 10 multiple items are introduced where some often go beyond the scope of the UX designer like security for example. However the writers of this research think that these items can serve as user requirements for the UX designer, he can now say to the developer which aspects of the cloud platform are essential from a user perspective. Perhaps some of them can be generalized to other professional software, for example that the cloud platform must be customizable would not only be an requirement for cloud platforms but for all professional software.

From the information presented in the previous three sections, ten design guidelines were extracted and created for UX designers to use when redesigning technical complex systems. These guidelines are (especially) applicable when the designer has a lower level of technical experience or understanding regarding software applications. The same reasoning as before applies here. These guidelines were based on the information gathered, and it is very much possible more (general) guidelines could be identified when information is added to the knowledge pool.

8 Conclusion

This paper presented guidelines that can be used as tools for the UX designer to better understand and work with technical complex systems. These guidelines were based on literature research. This literature research firstly described what the term "Complex System" entails and what different aspects of these types of systems are. These aspects are discussed in the section which describes how these systems can be understood from a non- or low-technical point of view. By decomposing and abstracting the system into smaller sections, aided by the FBS framework a better understanding of the system is creating by dividing the system into smaller segments. This could then be visualized using Stock and Flow diagrams. After complex systems had been elaborately discussed, another important aspect was described: communication. This provides the information necessary for the designer to re-design systems. The communication is improved by in-person communication, a clear division of roles and organizing group activities for the project group to partake in. By enhancing the interdisciplinary communication withing these groups, more mutual understanding about the design task is obtained and better products are delivered. This information gathered from this increased communication can them be applied to certain design principles which were described next, in order to finalize the process. This paper therefore described how to understand complex systems, how to gather information and how to apply this into a red-design according to design principles. This flow and its elements were then summarized into ten design guidelines for UX designers to follow while re-designing technical complex systems.

8.1 Future work

As stated in the discussion, there is more research to be done regarding the specifics of complex system design. Future work could therefore focus on gathering more information from which the presented guidelines could be improved or added to, or new guidelines could be created altogether. More researches could also be done on improving interdisciplinary communication. In the era of COVID-19, in-person communication became difficult and many people prefer to work from home. Also some companies have chosen to divide their teams across different locations, this means that it possible for example that the UX designer who is based in Amsterdam has to work together with an development team that is based in Argentina. Still, face-to-face conversations could happen online in video conferencing platforms such as Zoom. Technologies such as Cloud-Based Multiuser Virtual Reality, where organized group meetings are held virtually using VR technology, could replace real life face conversations or even improve communication efficiency. Since machine learning is often used as an optimization tool, optimization on how to schedule meetings, how to evaluate the outcome of onboarding tasks, how to set up a collaborative workflow, and how to distribute workload among the teams could be done with the help of artificial intelligence. More research could be done on how to use newly developed technologies to improve communication in the interdisciplinary workspace. When more time was given the authors of this research would also like to investigate other scientific

areas that have an impact on understanding complex system. Promising areas to investigate are psychology and pedagogy.

References

Adam Belloum. [n.d.]. Cloud Computing Cloud Platforms.

Adrian Bridgwater. 2021. Why Is Cloud Computing So Complicated?

Alexander Jones and Dr Volker Thoma. [n.d.]. Determinants for successful Agile collaboration between UX designers and software developers in a complex organisation. ([n. d.]).

Hakam W. Alomari, Vijayalakshmi Ramasamy, James D. Kiper, and Geoff Potvin. 2020. A User Interface (UI) and User eXperience (UX) evaluation framework for cyberlearning environments in computer science and software engineering education. *Heliyon* 6, 5 (5 2020), e03917. https://doi.org/10.1016/j.heliyon.2020.e03917

Bernhard J. Angerhofer and Marios C. Angelides. 2000. SYSTEM DYNAMICS MODELLING IN SUPPLY CHAIN MANAGEMENT: RESEARCH REVIEW. Brunel, 342–351.

Brian Stanton, Karuna Joshi, and Mary Frances Theofanos. 2015. Framework for Cloud Usability. In *Conference: HCI International 2015At: Los Angeles, CA , USA*. Los Angeles, 664–671.

Mary W. Chaffee and Margaret M. McNeill. 2007. A model of nursing as a complex adaptive system. *Nursing Outlook* 55, 5 (9 2007), 232–241. https://doi.org/10.1016/j.outlook.2007.04.003

Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. 2022. The Characteristics of Cloud Computing. *2010 39th International Conference on Parallel Processing Workshops* (2022), 275–279.

Cindy E. Hmelo, Douglas L. Holton, and Janet L. Kolodner. 2000. Designing to Learn About Complex Systems. *The Journal of the learning sciences* (2000), 247–298.

Cindy E Hmelo-Silver. [n.d.]. Design principles for Scaffolding technology-based inquiry. In *Collaborative Learning, Reasoning, and Technology*. Chapter 7.

Cindy E. Hmelo-Silver and Roger Azevedo. 2006. Understanding Complex Systems: Some Core Challenges. *The Journal of the Learning Sciences* 1 (2006), 53–61.

Cindy E. Hmelo-Silver, Surabhi Marathe, and Lei Liu. 2007. Fish Swim, Rocks Sit, and Lungs-Breathe: Expert–Novice Understandingof Complex Systems. *THE JOURNAL OF THE LEARNING SCIENCES* (2007).

Davide Bolchini and John Mylopoulos. 2003. From Task-Oriented to Goal-Oriented Web Requirements Analysis. *Fourth International Conference on Web Information Systems Engineering* (2003).

Wanchun Dou, Lianyong Qi, Xuyun Zhang, and Jinjun Chen. 2013. An evaluation method of outsourcing services for developing an elastic cloud platform. *The Journal of Supercomputing* 63, 1 (1 2013), 1–23. https://doi.org/10.1007/s11227-010-0491-2

Gartner. 2019. Gartner Says Global IT Spending to Grow 1.1 Percent in 2019.

Isha Upadhyay. 2020. Top 15 Major Characteristics of Cloud Computing.

Jakob Nielsen. 1995. Usability Metrics Tracking Interface Improvements. *IEEE Software* (1995), 12–13.

Janet L Kolodner. 2002. Learning by Design Iterations of Design Challenges for better Learning of Science Skills. *Design Reseach on Learning environments* (11 2002), 338–350.

Kevin Sookocheff. 2022. What complex systems can teach us about building software.

Laura Plonka, Helen Sharp, Peggy Gregory, and Katie Taylor. 2014. UX design in agile: a DSDM case study . In *Open Research On 15th International Conference, XP 2014, Lecture Notes in Business Information Processing, Springer.line*. Lancashire.

Luke Wroblewski. 2001. DESIGN CONSIDERATIONS FOR WEB-BASED APPLICATIONS . *Proceedings of the 45th Annual Meeting of the Human Factors and Ergonomics Society. Santa Monica, CA: Human Factors & Ergonomics Society. 2001.* (10 2001).

M. Hasan Jamal, Abdul Qadeer, Waqar Mahmood, Abdul Waheed, and Jianxun Jason Ding. [n.d.]. Virtual Machine Scalability on Multi-Core Processors Based Servers for Cloud Computing Workloads . ([n. d.]).

Stanislaw Paul MAJ and Chompu NUANGJAMNONG. 2020. Using Cognitive Load Optimiztion to teach STEM Disciplines to Business Students. In *2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 428–435. https://doi.org/10.1109/TALE48869.2020.9368351

Marton Sakal. 2009. GUI vs. WUI Through the Prism of Characteristics and Postures. *Management Information Systems* (4 2009).

M.T. Hoogvliet. 2008. *SaaS Interface Design Designing web-based software for business purposes*. Ph.D. Dissertation. Rotterdam University of Applied Sciences, Rotterdam.

Nicholas R. Jennings. 2001. Why agent-oriented approaches are well suited for developing complex, distributed systems. *Communications of the ACM volume 44* (2001), 35–41.

Qian Yang. 2018. Machine Learning as a UX Design Material: How Can We Imagine Beyond Automation, Recommenders, and Reminders? *AAAI Spring Symposium Series: User Experience of Artificial Intelligence* (3 2018).

Kurt A. Richardson and Michael R. Lissack. 2001. On the Status of Boundaries, both Natural and Organizational: A Complex Systems Perspective. *Emergence* 3, 4 (12 2001), 32–49. https://doi.org/10.1207/S15327000EM0304{_}3

David Sanderson, Jack C. Chaplin, and Svetan Ratchev. 2019. A Function-Behaviour-Structure design methodology for adaptive production systems. *The International Journal of Advanced Manufacturing Technology* 105, 9 (12 2019), 3731–3742. https://doi.org/10.1007/s00170-019-03823-x

Thomas Binder, Andreas Vox, Salim Belyazid, Hördur Haraldsson, and Mats Svensson. [n.d.]. Developing system dynamix models from casual loop diagrams. ([n. d.]).

Thomas Memmel, Harald Reiterer, and Fredrik Gundelsweiler. 2007. Agile human-centered software engineering. Lancaster.

Leigh Winowiecki, Sean Smukler, Kenneth Shirley, Roseline Remans, Gretchen Peltier, Erin Lothes, Elisabeth King, Liza Comita, Sandra Baptista, and Leontine Alkema. 2011. Tools for enhancing interdisciplinary communication. *Sustainability: Science, Practice and Policy* 7, 1 (4 2011), 74–80. https://doi.org/10.1080/15487733.2011.11908067

Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. 2018. Investigating How Experienced UX Designers Effectively Work with Machine Learning. In *Proceedings of the 2018 Designing Interactive Systems Conference*. ACM, New York, NY, USA, 585–596. https://doi.org/10.1145/3196709.3196730

Ying Zhou, Zheng Li, and Yingxin Li. 2021. Interdisciplinary collaboration between nursing and engineering in health care: A scoping review. *International Journal of Nursing Studies* 117 (5 2021), 103900. https://doi.org/10.1016/j.ijnurstu.2021.103900

The Impact of Edge and Fog Computing on IoT Applications for Smart Homes

Okke van Eck
11302968
University of Amsterdam
Amsterdam, The Netherlands
okke.vaneck@student.uva.nl

Florian Gerlinghoff
14092824
University of Amsterdam
Amsterdam, The Netherlands
f.gerlinghoff@student.vu.nl

Anne-Ruth Meijer
10978046
University of Amsterdam
Amsterdam, The Netherlands
anne-ruth.meijer@student.uva.nl

Abstract

Home automation can improve the lives of residents by monitoring and controlling several home appliances. However, the sensors needed for these services produce a significant amount of data. Storing and processing all these data could become the bottleneck of an applications. Edge and fog computing are promising compute paradigms that could help to overcome this challenge. In this literature review, we study 14 different home automation applications that use edge and fog computing and distill the advantages and disadvantages that come with these new approaches. The most often mentioned benefits are a lower latency, reduced network traffic and improved privacy and security, all thanks to the local processing of data. Other benefits and challenges are often dependend on the context of a surveyed application.

1 Introduction

The Internet of Things (IoT) is far from being a new thing. Already in the 1990s, Mark Weiser of Xerox PARC [27] wrote about the vision of ubiquitous computers blending with the environment. But even though most enabling technologies like the internet, embedded devices and low-cost connectivity technologies have existed for decades now, Corcoran [7] argues that only with the recent advent of cloud computing, IoT has found wide-spread adoption.

One particular field of application for IoT is home automation: sensors and actuators constantly monitor and control home appliances, thereby improving the lives of the residents of the *smart home* [1, 12]. One challenge that emerges is that all these sensors produce data constantly [26, 19]. Low-budget IoT devices themselves are not capable of handling this massive amount of data; therefore, it has to be sent for processing to the cloud [1]. This, however, leads to potentially higher latency, higher bandwidth requirements and higher energy consumption [31], and the data transport becomes the new bottleneck [25].

Especially for application domains with low latency requirements like gaming, *edge computing* was proposed to tackle these problems [5]. In this paradigm, some of the computation and storage is done on devices between the data sources and the cloud data centers and close to where the data originate [5, 25].

We want to investigate what edge and fog computing have to offer for applications in the field of smart homes and why researchers and developers have chosen or should choose it. In particular, we are interested in how it helps to process large amounts of data. We therefore pose the following research question:

> **RQ** What advantages and disadvantages does the use of edge computing and/or fog computing for big data smart home applications entail?

We divide **RQ** into the following three sub-questions:

> **RQ1** Which applications in the field of smart homes make use of edge computing and/or fog computing?
>
> **RQ2** What benefits did researchers or developers of the applications in **RQ1** expect when they decided to use edge computing or fog computing? In other words, *why* did they use edge computing?
>
> **RQ3** What drawbacks and challenges did edge computing and fog computing bring along for the applications in **RQ1**?

To answer these questions, we conducted a systematic literature review. The remainder of this paper is structured as follows. In Section 2, the terms Internet of Things, big data and smart home are explained in more detail. We further elaborate on some of the challenges that come with these technologies. Lastly, we describe edge and fog computing and what distinguishes these two approaches. In Section 3, we outline the procedure for this literature review, define inclusion and exclusion criteria, and give a short overview of the included material. In Section 4 we give a brief overview of what an IoT ecosystem looks like with edge and fog nodes. The other findings of the literature review are described in Section 5. We divided this section by application domain and present advantages and challenges of edge computing for each. In Section 6, we discuss our results, and finally conclude our paper in Section 7.

2 Background

In this section, we give an overview of the relevant background information for the research. Internet of Things (IoT) is described in relation to the home. The challenges of big data are listed and the relation of big data with cloud computing is detailed. After this, the cloud computing methods of edge computing and fog computing are explained.

2.1 Internet of Things

The *Internet of Things* (IoT) connects many small computers to each other, which then can exchange data over the internet [3]. These computers can be, for instance, dedicated electronic devices or integrated into objects of the everyday life. IoT allows the real world to be merged into the virtual world. An *IoT ecosystem* is a system that combines the components of IoT and tries to manage those in an efficient manner [3]. IoT applications can be categorized as consumer-based, industrial-based and infrastructure-based [3]. Consumer-based applications include smart wearables, smart vehicles and smart homes. Industrial-based applications include smart manufacturing processes and automation of industrial tasks. Infrastructure-based applications are, for example, smart cities and smart environments. Smart homes and home automation are our main focus for this work [3].

2.2 Home automation and smart homes

The terms *home automation* and *smart home* both describe the concept of automating services for residents of a home. However, they cannot be considered synonyms. *Smart home* refers to the residence itself that includes smart technologies to improve the quality of life of its users [18]. Home automation, on the other hand, means controlling and automating home appliances and domestic features via the network or a dedicated remote control [13]. The term smart technology is often used in the context of home automation and smart homes. Smart technology has the ability to gather data from its surroundings and act based on that data [18]. One example is a smartwatch: it can, for example, measure how loud its surroundings are and warn its owner of possible damaging levels.

There are various benefits of creating a smart home, but challenges exist as well. We discuss them in Section 2.2.2 and Section 2.2.3, respectively.

2.2.1 Components of smart homes

Smart homes consist of various devices and sensors, making up their own IoT ecosystem [32]. The components of a smart home can be divided into three categories: cyber-physical, connectivity and context-aware. The first category, cyber-physical, is comprised of the actual smart devices that are installed directly in the home. They are responsible for the measuring, changing and operating the home. Examples include smart appliances, energy management systems and smart charging points. These components need to be able to communicate among themselves and contact the outside world. This is the responsibility of the second category of components. The connectivity allows devices to communicate through an IoT gateway. and the homeowner to interact with the smart technology through, for example, a mobile app. The last category are context-aware components. These are in charge of the decisions in the smart home; they provide the intelligence. This category also manages policies and security configurations created by the residents. The intelligence includes activity recognition, event detection, behavioral analysis and predictions. These aspects are mostly placed in cloud computing systems, including edge or fog computing.

2.2.2 Benefits

Extending a home with smart technology can provide immediate advantages and long-term benefits [18]. The potential benefits can be divided into health-related, environmental, financial and social benefits. Literature reviews on the topic show that the most common benefit is related to improved health, with environmental being second [18]. Less mentioned benefits are the financial and social benefits.

Smart homes can provide support to vulnerable people inside the house [18]. Smart homes have multiple functions for the health of the residents. The technology can aid monitoring, managing and consulting. For example, the hearth rate of users can be monitored and the user can be alerted of irregularities. A chronic disease can be helped with by managing prescriptions and registering data. In terms of consulting, smart home technologies can for example enable virtual visits of medical professionals.

Environmental benefits are provided by smart technologies focused on energy efficiency [18]. With climate change and global warming, the interest in energy-efficient devices has increased. Technologies that monitor energy consumption, control the consumption and aim at energy optimization allow homes to become more energy-efficient.

The financial benefits can be associated with the health-related and the environmental benefits [18]. A benefit of improving energy efficiency is the reduction of energy consumption. A reduction in energy consumption will lead to a reduced energy invoice. An effect of monitoring energy consumption is the transparency it can provide. The monitoring allows users to compare energy costs from varying providers to choose the financially optimal provider. Health-related applications for the smart home could result in prolonged home care, which could be financially beneficial. Summarizing, smart homes provide multiple financial benefits. However, studies showed that the investment and prospected maintenance compared to the relatively low prospected saving causes financial benefits to not result in users choosing to switch to a smart home [18].

Smart homes can provide social benefits by avoiding users to become isolated [18]. The technology can aim to provide support and establish contact with loved ones. However, this benefit is not always supported. By replacing face-to-face contact, the feeling of isolation could also increase.

2.2.3 Challenges

There are multiple challenges that hinder the adaptation and implementation of smart homes [18]. A significant challenge is the technology on which smart homes are based.

The technology used is an important factor of not implementing smart homes [18]. The challenges in the technology are mainly focused on security, privacy, reliability, complexity and finance. The ability of gathering massive amounts of often private data by the smart technologies raises concerns about security. For users, it is often unclear what protocols are used to store their information. Moreover,

the applications need to be secured at the *device layer*, *network layer*, *cloud layer*, and *human layer* [10]. This requires a lot of knowledge, energy and time to be accomplished.

Another challenge is the reliability. The technology needs a decent life cycle to justify the investment. Financial challenges for adopting smart home technology are the initial cost and maintenance cost [18]. This challenge includes the lack of understanding how environmental and healthcare related devices could benefit financially.

2.3 Big data and its challenges

IoT smart home systems include components that collect a lot of data—big data—to base their decisions on. This results in a time-demanding and challenging task [14]. Big data refers to a volume of data that is difficult to analyze, process and store with traditional technology [14]. The use of big data provides opportunities in decision-making tasks, while introducing challenges as well. The challenges consist of data storage, transmission, management, processing, and analysis [30].

Data storage The volume, velocity and variety of big data provide a challenge for storage [30]. The velocity can cause the data to quickly scale up, which is difficult to support with traditional methods for data storage. Cloud computing is an alternative solution for data storage and can support the velocity, variety and volume better than traditional methods. However, cloud computing could be a cost-inefficient solution.

Data transmission Using big data means that the data needs to be transmitted on multiple occasions [30], for example, when it is collected, aggregated from multiple locations, managed and analyzed. Transferring high volumes of data multiple times introduces a large time demand. Therefore, it is important to use techniques to minimize the transferred amount.

Data management To efficiently manage data, it needs to be cleaned, stored and organized. However, on big data, this can be a challenging task [30]. the costs of sensors and computing has reduced, which makes it easier to collect big amounts of data in a short time [10]. Technologies, for example Hadoop, are available to provide management solutions. However, there is still the challenge of automatically generating metadata to describe the data [30]. This is a challenge due to the complexity of the data and the variety.

Data processing In order to process large amounts of data, a lot of computing resources are required [30]. Traditional computing solutions do not provide enough computing power, but cloud computing provides one solution. Another challenge while processing data is data locality. This is the process of performing the computations on the machine where the data resides instead of moving the data.

Data analysis Extracting information from big data is an important factor of working with big data. However, it does impose challenges. It requires complex algorithms and parallel processing. The algorithms need to be able to scale according to the data and to possibly cope with diverse data.

2.4 Big data with cloud computing

As stated before, big data brings a lot of challenges. Its large volume and high throughput require special technology to benefit from big data through data-driven applications [20]. Data warehouses were the first solution to store relational data and process them efficiently. Due to their bad support for unstructured data and high cost, data warehouses were replaced by distributed file storage and processing platforms like Hadoop [20]. Hadoop was one of the first to provide reliability and scalability in data processing. Moreover, the platform was able to store and analyze unbounded big data on clusters. Nowadays, new platforms include more cloud technology to provide cheap, flexible and scalable big data solutions [20]. Many IoT scenarios today require collection of data from millions of devices, while processing the data at the lowest response time possible for creating analytics [16].

Big data analytics systems are latency-sensitive and can create petabytes of data [22]. This amount of data is impractical to stream back and forth between the cloud data centers and end-devices. This issue stimulated the development of two new extensions on cloud computing: *edge computing* and *fog computing* [22]. Both are discussed in the next section.

2.5 Edge and fog computing

Two compute paradigms that solve the latency and bandwidth problems with cloud computing are *edge computing* and *fog computing*. Modern day cloud computing applications generate loads of data, which would be very costly to move and process all in the cloud [10]. Edge computing was introduced to aggregate all data from the IoT devices and send them efficiently to the cloud. It pre-processes the data and eliminates redundancy, so the data size is reduced without affecting the information content too much [15]. The cloud itself can then decide how to use the delivered data.

Fog computing is a layer in between the cloud and edge computing. It was introduced to further reduce the data by executing several cloud functions, like further extensive pre-processing, pattern mining, classification, prediction, and visualization [32]. Fog computing nodes handle these functions and provide quick analytics of the collected data while sending the aggregated results to the cloud. Data arriving at the fog computing layer can be unstructured and does not have a predefined model. After applying the necessary computations, the fog compute nodes send structured data to the cloud, possibly containing privacy-sensitive data. This makes the fog nodes very vulnerable to various attacks, more than edge computing [32].

Note that in the literature, many authors use different definitions for fog computing. A study often does not make the distinction between the edge computing functionalities and fog computing functionalities. Instead, they simply state that the application makes use of fog computing in general, while their definition of fog computing includes functionalities from our definition of edge computing.

3 Method

To answer our research question **RQ**, we strove to compile a comprehensive overview of smart home applications that adopted edge computing and understand the motivation and reasoning behind the use of this paradigm. Therefore, we conducted a systematic literature review [17]. Compared to other methods for literature studies, this one allows for a guided traversal of the available literature, can reduce bias because of its systematic approach, and has a clear criterion for when to stop the search.

The last point made it possible for us to reduce the scope of our search and in this way the number of research articles to review, which was necessary because of the upper page limit of this work as well as limited time. In particular, we only considered documents that discuss how edge and fog computing can help with the processing of large amounts of data.

To find relevant articles, we used the Scopus[1] search engine. Again because of the page and time limit, we did not consider other search engines since the Scopus search already yielded enough articles to review. The following search string was used:

```
TITLE-ABS-KEY ( "edge computing" OR "fog computing" ) AND
TITLE-ABS-KEY ( "big data" ) AND
TITLE-ABS-KEY ( home AND ( automation OR smart ) )
```

We included research articles, conference papers, and other literature surveys, all in English language, in our review. That we also cover other secondary studies is in contrast to the recommendation in [17], but we concluded that surveys, too, can help to answer our research question. The 37 search result were filtered further according to the inclusion and exclusion criteria shown in Table 1. In the end, this literature review covers 14 documents spanning the period from 2018 to early 2021 (see Figure 1). Each article was selected, reviewed and summarized by the same single person. The summaries were then categorized by application type and synthesized. The quality of studies was not considered explicitly.

[1] `https://www.scopus.com/`

Table 1: Inclusion and exclusion criteria for the literature review

Inclusion Criterion	Exclusion Criterion
• The document describes one or several smart home application that uses edge or fog computing OR • The document describes a technical solution and a smart home application is used as case study	• The article does not describe a concrete smart home application OR • The authors do not explain why they used fog computing or what the advantages and disadvantages of this approach are

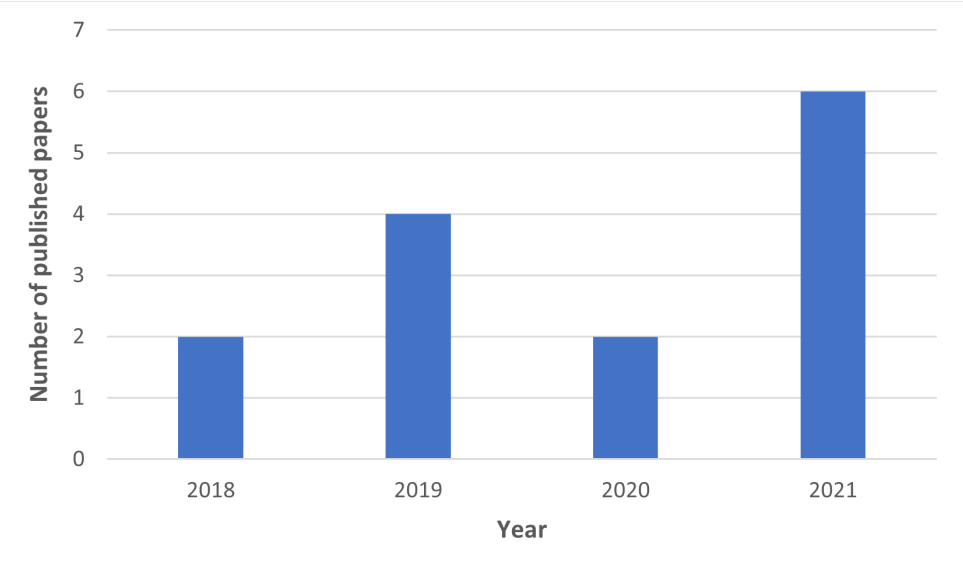


Figure 1: Number of papers included in this review, per year published

4 IoT ecosystem and architecture

An ecosystem that revolves around IoT applications historically consists of three layers: IoT Device Layer, IoT Fog Layer, and IoT Cloud Layer [10]. We found that nowadays, some applications also involve edge computing in the fog layer. However, edge computing is fundamentally different from fog computing, which is why we add the IoT Edge Layer to our definition, which resides between the Device Layer and Fog layer. Figure 2 shows the interaction between the four layers. Credits for this image should go to Farahani et al. [10], as we took figure 4 from their paper and added the edge layer to it.

We shortly cover each of the layers to provide an understanding of what the ecosystem entails. This overview will be very concise, as it is our goal is to highlight the impact of edge computing and not the entire IoT ecosystem.

Device layer The device layer is a set of smart IoT devices that enable individual users to collect data for their usage. These devices can be classified as either physical sensors or virtual sensors [10]. Physical sensors are hardware that perform measurements, while virtual sensors come in the form of software and mobile applications. These devices use a connectivity and communication protocol based on the requirements of the application.

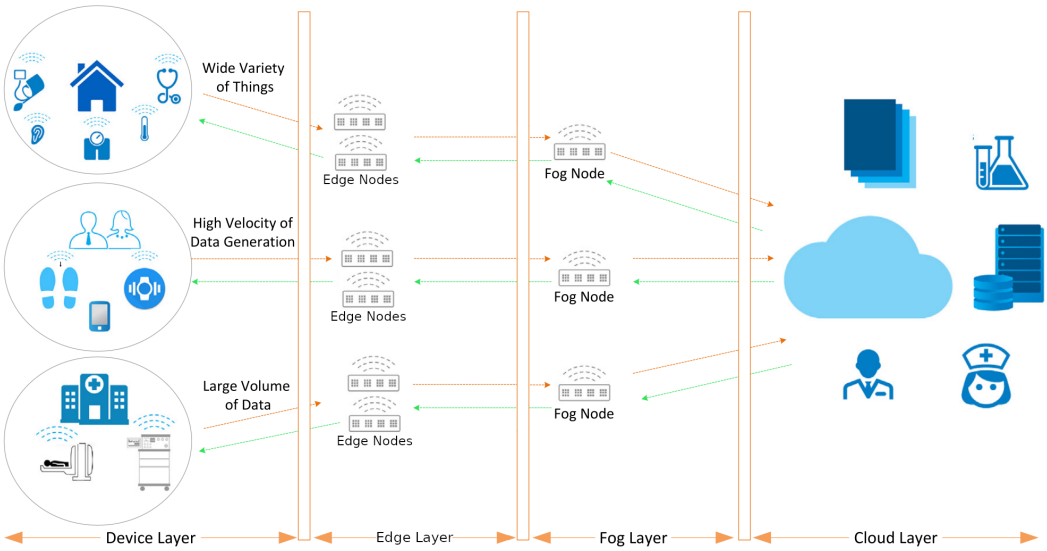


Figure 2: Four-layer IoT architecture (with changes from [10]).

Edge layer The edge layer pre-processes the generated data from the device layer and eliminated redundancy. It limits the data size without affecting the information content too much [15]. This reduced dataset is then forwarded to the fog layer.

Fog layer The fog layer handles multiple application specific functions. It can further reduce the data by for example performing pattern mining, classification, or prediction [32]. Moreover, the fog layer can also be used to create visualizations that are used to determine data-relevance. The goal of the fog layer, for performance critical applications, is to reduce latency and bandwidth usage as much as possible [10].

Cloud layer Lastly, the cloud layer is where the fog node data comes in and is processed by the application. The cloud is often a multi-layer architecture, and consists of a *connectivity layer*, a *user/device/data management layer*, and an *application service layer* [10]. The connectivity layer established connectivity between devices, fog nodes, and the cloud. The management layer aggregates the data from multiple sources and stores it locally. Lastly, the application service layer reads the data from storage and uses it for application specific purposes.

5 Edge and fog computing in smart home applications

With the relevant terms and concepts explained, this section will provide an overview of the big data smart home applications we came across during the literature review. These were categorized by their domain as either related to energy management, health or other applications that enhance smart homes ("smart living"). We will also emphasize the benefits and drawbacks/challenges that were mentioned by the authors. Note that we stuck to the terminology of the authors of each article, so different words might be used for the same concept in different paragraphs.

5.1 Energy management

All the IoT devices together consume a lot of energy, but they can also help with efficient energy management. In this section, we explore different applications that manage the energy consumption of smart homes on a small and on a big scale.

5.1.1 On a small scale

In [29], Xia et al. present an "edge-based energy management system" whose job is the scheduling of home appliances. It aims to create a schedule that maximally utilizes energy produced by the smart

home's solar energy system. The scheduling algorithm runs on an edge device. In their proof of concept, Xia et al. chose a Raspberry Pi 3B. The authors do not explain in detail why their system runs on the edge rather than in a central cloud data center, but they mention that they used a Raspberry Pi "due to its low-cost and appropriate computation capability" [29]. In other words, the system runs on the edge because it is good enough for this scenario. No cloud needed.

Energy management plays a role, too, in the real case scenario Di Martino et al. describe in [8]. They outline how users can set constraints and preferences for their home appliances. Based on these and other information like the weather forecast, an optimal schedule for energy consumption of the appliances is calculated. This calculation takes place on the fog nodes, but it is important to balance the computational load between nodes. The authors describe in this paper an approach to parallelize the calculation and to distribute the load. This task also includes finding a hardware allocation such that available resources are optimally utilized.

With their approach, Di Martino et al. want to minimize the need for cloud servers. As advantages of the fog computing paradigm they mention that 1) security issues are avoided because data is not stored remotely, 2) responses can be provided fast and reliably, 3) results can be delivered even if the internet connection fails as long as the local network stays intact, and 4) fewer data is transferred over the network, therefore reducing the likelihood of congestion.

5.1.2 On a larger scale

Singh and Yassine describe a platform in [26] for analyzing data about the energy consumption from multiple home appliances. The goal is to find patterns in and possibly make forecasts about consumers' energy consumption, which enables an optimized operation of smart power grids. The authors start their design process from the requirements for this platform. These include the ability to obtain near real-time responses from the system. The authors use fog computing to satisfy this requirement. Another mentioned advantage of fog computing is that these nodes allow for resource-efficient computations, which can reduce energy consumption and cost.

However, Singh and Yassine mention that the fog computing nodes fall short for very computationally intensive tasks or when access to a huge amount of data has to be stored. Hence, they combine fog computing with the traditional cloud computing paradigm. Fog computing nodes pre-process data collected from IoT devices and send them to the cloud, where data from multiple fog computing nodes is aggregated, stored, and analyzed. That is, tasks that are not latency-sensitive can be offloaded to the cloud.

The authors also mention briefly that privacy and security issues arise when data from multiple households is processed on the same fog node. The same problem exists when processing the data in a cloud data center, though.

A similar design that distributes tasks between the fog nodes and the cloud was chosen by Pham et al. in [22]. Therein, they present the use case of an energy management system for multiple smart homes within a district. Big data analysis is applied, among other purposes, to detect unusually high energy consumption or make predictions about the consumption. The authors argue that for time-critical tasks such as anomaly detection, the computation should take place on fog nodes. Moreover, these nodes pre-process data before it is sent to the cloud data center. This way, the amount of transferred data and therefore the required bandwidth are reduced significantly. The cloud is responsible for handling computationally expensive tasks as well as batch-processing tasks that need access to the big cloud storage, like the consumption prediction.

One of the biggest challenges is what Pham et al. call *elasticity*. When a complex application is divided into components that are distributed across the cloud, fog, and end-devices, it must still be possible to migrate components vertically between the fog and the cloud and horizontally between different fog nodes, according to the authors. When such a migration must happen can depend, for example, on the number of end-devices connected to a single fog node. Horizontal migration might also be required when end-devices change their physical location. Components are then moved to a fog node that is geographically closer to the end-device.

Another issue the authors raise is vendor lock-in. When components must be able to migrate between the cloud and different fog nodes, each potentially from another vendor, compatibility must be ensured. Lastly, all components must be able to interact if required, regardless of whether they are deployed on end-devices, fog nodes or in the cloud. After describing all these challenges, Pham et

al. go on to propose a framework for IoT big data analytics applications that provides the required elasticity.

Butt et al. sketch out a model for a global energy management system in [6]. It has four layers. Layer 4 consists of brokers, each responsible for a number of buildings that contain smart meters. Smart meters send requests for energy to a broker. The broker communicates with the cloud (layer 1) and, based on the information it receives, selects one of several fogs in the corresponding region. Fogs comprise the second layer in this model. The broker installs a specific application, a so-called *cloudlet*, on a virtual machine in the selected fog. This cloudlet has to locate the nearest microgrid, i.e., a grid that spans only a few energy consumers and producers like solar panels, that has a surplus of energy and can additionally supply the smart meter that initially made the energy request. If no such microgrid can be found, then the energy must come from the country's macrogrid. The microgrids make up the third layer in this model.

In [6], the purpose of using fog computing is to achieve scalability by distributing the load from one central cloud to many fogs. In addition, improved latency and security are mentioned as benefits of fog computing. Load balancing within the fogs is still an issue, for which Butt et al. go on to simulate and compare different algorithms.

5.2 Health-related applications

Traditional healthcare is revolved around citizens visiting doctors for their care [10]. This hospital-centric model requires patients with chronic diseases to make frequent visits to hospitals or clinics in order for doctors to make observations. Some other challenges and barriers of the traditional model are: scaling issues with the increasing number of patients, the increasing average age of the population, urbanization, shortage of healthcare workers, and rise of medical costs [10]. These challenges and barriers were a driving force behind a new IoT-driven healthcare model, which transitions from hospital-centric to patient-centric healthcare.

Patient-centric healthcare entails that the received healthcare is tailored to someone's individual needs [10]. In order to integrate hospitals or clinics with patients, it is required to utilize IoT applications. Examples of areas in which IoT applications can play a role are: hospitals, clinics, mobile-clinics, telemedicine, wellness, smart homes, and smart cities. In this section, we will cover health-related applications that are integrated in smart homes.

5.2.1 Patient monitoring

One application to improve the health of elderly people is a monitoring platform proposed by Alexandru et al. in [2]. It uses data from wearables, smartphones and home sensors to collect important information about patients, which enables a patient-centric healthcare. As an example, motion sensors can determine if a person has fallen and needs help. These devices make up the *data generation layer*. Data is then sent to the *fog layer* for processing and small analysis, and then possibly to the *cloud layer* for storage and deeper analysis. The fourth layer is the user interface to various stakeholders.

The requirements for the platform of Alexandru et al. include being able to timely process a large amount of data and integrating with a variety of devices. Fog computing is a good solution because (1) it allows for real-time processing of data for time-critical applications due to lower latency, (2) it can be geographically distributed and scaled, (3) it is capable to provide privacy and security, and (4) fog nodes supports different communication standards, so they can exchange data with various devices and sensors.

The authors also list a lot of other benefits of fog computing. These include reduced network traffic because fog nodes can filter and pre-process data, support for mobility and low energy consumption since no additional cooling system is needed. Because of the resource constraints of the fog nodes, the authors offload heavy tasks and storage to the cloud. Other challenges that come with fog computing are that fog nodes must support a lot of communication technologies and protocols, flexible naming and name resolution mechanisms must be deployed, and resource managers must consider the context of the fog-cloud environment.

Feng et al. [11] build their solution based on a similar layered architecture with fog nodes in the middle. They propose a platform for the analysis of health information in hospitals. According

to the authors, it can also be used to support smart homes. Their main concern is to reduce the energy consumption of the whole solution. This requires energy-efficient sensor networks, but also an approach to minimize the energy consumed by fog nodes, which are most of the time busy with data processing. They describe their clustering approach in their paper. In a simulation, they then show that using their fog computing platform can contribute to saving energy when compared to traditional cloud computing.

The other reason for choosing fog computing in [11] is the reduced latency, which is important for delay-sensitive healthcare applications. Hence, fog nodes analyze sensor data, process important requests and events immediately, and send the less time-critical data to the cloud layer, where it is further processed and stored. Again in a simulation, the authors show that this setup reduces network delay, congestion and round-trip time compared to the cloud-only scenario.

For Bera et al. [4], data integrity is important to ensure correct predictions when applying big data analysis to COVID-19-related data collected by a home monitoring system. In particular, they want to avoid data poisoning attacks when there are untrusted participants. Additionally, confidential medical data should only be visible to authorized parties. In their system model, medical devices and wearables send data to a fog server, which extracts useful information that are relevant for the predictions. When fog nodes do not have to trust other fog nodes, it is possible to let them be managed by different entities. The others propose the use of a private blockchain to establish trust in the provided data. Fog nodes encrypt the information they extracted and send them to a cloud layer, which is responsible to add them to the blockchain. The prediction algorithm is able to decrypt the data and can use them while being certain that the data originated indeed from a real medical device.

Fog computing is used in this model because the sensitive medical data should only be processed either locally or by the trusted big data analysis algorithm. For the data transfer, the data is encrypted. Other benefits of fog computing, according to the authors, are reduced latency and better load distribution among the many fog nodes.

In the paper, the authors only vaguely describe their use case and mostly in abstract terms. Furthermore, it is questionable if trust between the fog nodes, which in the authors' model run in the same local network, cannot be established in a less expensive manner.

5.2.2 Assisted living

Privacy is a big concern for Wu et al. In [28], they describe a *nursing system* which recommends activities to elderly people. For example, it is possible to receive TV show recommendations or automatically order food. Multiple smart homes communicate with what the authors refer to as *edge node*. Depending on QoS requirements like latency and computational complexity, tasks are either processed on these nodes or in the cloud. Besides lowering the latency, using edge nodes also reduces the network traffic and leads to a more robust network, according to the authors. Edge nodes also act as a caching layer between the cloud and smart homes. Furthermore, they participate in the maintenance of a blockchain that is used to preserve the privacy of the users of the nursing system.

5.2.3 Generic health applications

In [9], Dong and Yao propose an IoT platform that could help with non-pharmaceutical interventions (NPIs) to control the COVID-19 pandemic. An architectural layer of low-end, decentralized fog computing nodes is responsible for handling time-sensitive NPIs like *quarantine monitoring*. Fog computing is suitable for these kinds of tasks because the nodes are located closer to the data source and the latency is therefore lower. Furthermore, location-sensitive NPIs like *contact tracing* are implemented in the fog layer. The fog nodes must therefore be able to track the location of connected IoT devices and support their mobility.

NPIs that require a lot of computing resources or a lot of data are offloaded to the cloud layer. For example, *outbreak forecasting*, which uses machine learning algorithms, is handled by this layer. According to Dong and Yao, a proper strategy has to be found to decide if computing resources are allocated at the fog or cloud layer. Such a strategy could help to reduce power consumption while still satisfying time constraints. The authors propose a "smart gateway" that makes this decision based on parameters like the data that needs to be processed, the current network congestion and energy consumption.

Shehab et al. [24] consider fog computing to be an important part of the telehealth ecosystem, too, because health-related applications are often time-critical and privacy-sensitive. The problem, however, is that fog nodes are often geographically distributed, have only few computing resources available and can fail anytime. Current stream processing platforms are not able to handle to manage fog resources efficiently. For this reason, the authors propose a new resource manager specifically designed for fog computing. It enables telehealth applications that are fault-tolerant scalable.

Pani et al. point out in [21] that admittedly the response time plays a crucial role for the processing of emergency health data, but the energy requirements of IoT devices should be considered, too. For this reason, they propose to categorize tasks as either delay-sensitive or delay-tolerant. To reduce energy-consuming communication, only delay-sensitive packets are sent immediately to upper layers (fog/cloud). Delay-tolerant packets are only forwarded as batch of a size bigger than a certain threshold. The authors state that benefits of fog computing in general are lower latency, higher bandwidth, and less geographic dispersion.

5.3 Living in smart homes

Some of the surveyed applications do not fit into the energy or health category. These are summarized in this section.

Kireev et al. [16] describe a system that monitors the state of utilities in the home, such as water inlets or pumping stations. It can detect and classify faults and then notify operators or residents. Furthermore, it can make forecasts about the utilities, which can, for example, be used to schedule repairs. The authors only describe their proposed system, but do not evaluate it empirically.

In this system, IoT devices called *controllers* regularly collect data from the sensors in the house and, after an initial filtering, send them to a *router* which resides in the same network. The router compares the received data to thresholds for each sensor and, if the measurements are below or above some limit, it tries to classify the problem. In case of an emergency, the cloud server is called, which then can notify the responsible person. In the cloud, data from multiple routers are aggregated, stored and analyzed.

Thanks to the use of edge computing and therefore the highly distributed nature of this system, data from numerous sensors can be processed. Moreover, because data is first filtered and processed locally, less network bandwidth is required.

In [23], Ramirez-Prado et al. present a "smart floor" that can be used for movement and activity tracking of a smart home's tenants. They use the fog computing paradigm to reduce the amount of traffic to the cloud. This reduces "overhead such as time, latency and throughput, energy consumption, and cost" [23]. They also emphasize the importance of privacy, security and integrity of the sensitive health data and suggest fog computing is a suitable approach in this regard.

5.4 Summary

Table 2 gives an overview of all the benefits and Table 3 of all the drawbacks/challenges that were mentioned in the papers included in this review.

The advantage that is most often mentioned—only two papers do not talk about it—is the reduced latency compared to traditional cloud computing thanks to the fog nodes being located closer to the edge devices and sometimes even in the same local network. This makes fog computing suited for real-time processing and time-critical applications. However, fog nodes mostly do not have large computational resources. Many authors propose to circumvent this problem by offloading resource-intensive tasks and the storage of data to the cloud. That is, they suggest to extend the cloud with fog computing, not to replace it.

Even this extension can reduce the network traffic to the cloud significantly, since not all data has to be processed there. When fog nodes reside in the local network, it can also improve the reliability of the whole system because it is now more resilient against internet outages: even if cloud data centers cannot be reached, fog nodes can do some of the processing.

Furthermore, not having to process data remotely can greatly benefit the privacy and security of the application. This seems to be especially relevant for health-related applications. However, not every fog layer is located at the local network. Some solutions use fog nodes that are closer to the

Table 2: Benefits of using edge computing for smart homes mentioned in surveyed papers

Domain	Ref.	Latency	Network traffic	Reliability	Privacy/Security	Scalability	Mobility	Interoperability	Energy efficiency	Cost
Energy management	[29]	-	-	-	-	-	-	-	-	✓
	[8]	✓	✓	✓	✓	-	-	-	-	-
	[26]	✓	-	-	-	-	-	-	✓	✓
	[22]	✓	✓	-	-	-	-	-	-	-
	[6]	✓	-	-	✓	✓	-	-	-	-
Health	[2]	✓	✓	-	✓	✓	✓	✓	✓	-
	[11]	✓	✓	-	-	-	-	-	✓	-
	[4]	✓	-	-	✓	✓	-	-	-	-
	[28]	✓	✓	✓	✓	-	-	-	-	-
	[9]	✓	-	-	-	-	✓	-	-	-
	[24]	✓	-	-	✓	-	-	-	-	-
	[21]	✓	✓	-	-	-	-	-	-	-
Smart living	[16]	-	✓	-	-	✓	-	-	-	-
	[23]	✓	✓	-	✓	-	-	-	✓	✓

Table 3: Disadvantages and challenges of using edge computing for smart homes mentioned in surveyed papers

Domain	Ref.	Resource constraints	Data segregation	Resource management	Load balancing	Elasticity	Interoperability	Energy consumption
Energy management	[29]	-	-	-	-	-	-	-
	[8]	-	-	-	✓	-	-	-
	[26]	✓	✓	-	-	-	-	-
	[22]	-	-	-	-	✓	✓	-
	[6]	-	-	-	✓	-	-	-
Health	[2]	✓	-	✓	-	-	✓	-
	[11]	-	-	-	-	-	-	✓
	[4]	-	-	-	-	-	-	-
	[28]	-	-	-	-	-	-	-
	[9]	✓	-	-	✓	-	-	-
	[24]	✓	-	✓	-	-	-	-
	[21]	-	-	-	-	-	-	-
Smart living	[16]	-	-	-	-	-	-	-
	[23]	-	-	-	-	-	-	-

edge devices, but may process data from multiple smart homes. As in cloud data centers, it is then important to arrange for sufficient segregation of this data to avoid privacy issues.

The geographical distribution of fog nodes makes it possible to scale systems to support many smart homes with their edge devices, sensors and users, and it allows to distribute load between different nodes. Fog computing can help to deal with mobile users and devices that change their positions: when a device gets closer to another fog node, this node can take over the processing for this particular user.

But the distribution and the sheer number of fog nodes introduce several challenges, too. Firstly, a new strategy is needed to allocate nodes efficiently to users. It must also take into account that fog nodes can crash and might even crash often [24]. Similarly, new approaches for scheduling and load balancing have to be found that take the fog computing paradigm explicitly into account and achieve an efficient utilization of both fog nodes and cloud servers. Lastly, because systems can be comprised of many fog nodes and because some applications have to support mobility of users, new solutions have to be developed to migrate application tasks between fog nodes or between the fog and the cloud layer. Pham et al. [22] call this *elasticity*.

One advantage of fog computing mentioned in [2] is that fog nodes can communicate with many different edge devices because they can be built to support various communication technologies and protocols. This, however, is at the same time a challenge because integrating and standardizing these technologies is not trivial. [22] points out the need for compatibility between vendors.

The better energy efficiency compared to traditional cloud computing is mentioned as benefit of fog computing by three authors. On the other hand, it is also important to keep the energy consumption of the additional fog nodes in mind, according to [11].

For some authors, the low cost of the edge devices themselves and the whole solution is a benefit.

6 Discussion

The introduction of IoT in a home setting allowed for an enhanced home experience [3], but also resulted in enormous amounts of data produced. These datasets are often personalized, and can contain petabytes of data that needs to be processed [22]. The problems with processing big data is generally tackled through cloud computing [30], but this approach is not suited for latency-sensitive applications [20]. Streaming petabytes of data back and forth between cloud data centers and end-devices is undesirable, which motivated the introduction of edge and fog computing [22].

Our literature review suggests that edge and fog computing indeed can provide the quality of service required by latency-sensitive applications. Many papers are published that state that the reduction of the latency is the purpose of using fog computing. Since this approach is adopted by many authors and we have not read about qualified alternatives, we count this as evidence that fog computing is also in practice very beneficial for these kinds of applications.

Furthermore, the adoption does not seem to introduce many new challenges. Several papers even mentioned zero challenges or disadvantages. But readers should not conclude from this that edge and fog computing can be introduced without obstacles. A more decentralized distributed computing paradigm requires efforts in finding new solutions for difficult topics such as resource management and load balancing. It also requires to think about new programming models and how to place applications along the cloud-edge continuum. There are many open research questions in this emerging field of edge computing.

Many of the covered papers do not do a good job in discussing the technical decision to use fog computing. They just state *that* they are using fog computing and name a few benefits, but often without covering the negative consequences and challenges or exploring design alternatives. But this is a general pity with academic papers: that they mostly just describe the end result and not how the authors got there. As positive examples we want to highlight Sing and Yassine [26] as well as Alexandru et al. [2] who started their designs from the requirements.

When authors describe technical challenges of edge and fog computing, then often in the context of a solution they present for this particular problem. Of course, it is very nice that many of these new challenges are tackled, but we would appreciate a more thorough assessment of pros and cons.

One major problem we encountered during our study was the lack of shared definitions for common terms. The line between edge computing and fog computing is drawn differently by every author, and getting consensus alone on what exactly the "fog" is seems to be difficult. Fog computing can cover everything from a Raspberry Pi next to some smart devices to processing on the router at the local network (some authors refer to this as *mist computing*) to micro data centers that are geographically distributed.

We certainly did not cover all relevant papers in this survey. For example, we did only use one search engine. Nevertheless, we think that the papers we included are enough to uncover the main benefits and challenges of edge and fog computing in a smart home setting. More papers could probably reveal further niche use cases, advantages and disadvantages, but would most likely also focus on the main driving force behind the adoption of fog computing, namely the lower latency. Furthermore, we were constrained by page and time limits.

Initially, each reviewer independently inspected a subset of the papers in the search results. Later in the process, other reviewers took a look at the included papers, but did not evaluate excluded papers again. This means that the individual decisions of each reviewer carry a lot of weight. The reviewers also needed some time to synchronize their inclusion and exclusion criteria.

7 Conclusion

In this systematic literature review, we examined papers which describe home automation applications that use edge or fog computing to manage big data. The applications we found come from the domains of energy management, healthcare and enhanced smart homes. Most of these applications were time-critical, so traditionally cloud computing was not a suitable approach for data processing because of potentially high and variant latency.

Edge and fog computing provide a solution to this problem. Data is processed closer to the data source, potentially even in the same local network. This can reduce the latency significantly. But a cloud component is still required by most applications since fog nodes are often constrained in their computing power and storage capacity. Therefore, resource-intensive and time-uncritical tasks are often offloaded to the cloud. Still, having a fog layer between data sources and the cloud can help to reduce the network traffic, because fog nodes can filter and pre-process the data. Besides lower latency and reduced network traffic, seven other benefits were mentioned in some of the papers included in this literature review.

Some authors also mentioned disadvantages and challenges that come with edge and fog computing. Another seven of those we distilled in this review. Despite them, we conclude that edge and fog computing can keep their promise of enabling time-critical big data applications in smart home environments.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[2] A. Alexandru, D. Coardos, and E. Tudora. IoT-based healthcare remote monitoring platform for elderly with fog and cloud computing. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pages 154–161. IEEE, 2019.

[3] S. Bansal and D. Kumar. IoT ecosystem: A survey on devices, gateways, operating systems, middleware and communication. *International Journal of Wireless Information Networks*, 27(3):340–364, 2020.

[4] B. Bera, A. Mitra, A. K. Das, D. Puthal, and Y. Park. Private blockchain-based AI-envisioned home monitoring framework in IoMT-enabled COVID-19 environment. *IEEE Consumer Electronics Magazine*, 2021. Early Access.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing - MCC '12*, pages 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[6] H. Butt, N. Javaid, M. Bilal, S. Aon Ali Naqvi, T. Saif, and K. Tehreem. Integration of cloud-fog based environment with smart grid. In F. Xhafa, L. Barolli, and M. Greguš, editors, *Advances in Intelligent Networking and Collaborative Systems*, pages 423–436, Cham, 2019. Springer International Publishing.

[7] P. Corcoran. The Internet of Things: Why now, and what's next? *IEEE Consumer Electronics Magazine*, 5(1):63–68, 2016.

[8] B. Di Martino, S. Venticinque, A. Esposito, and S. D'Angelo. A methodology based on computational patterns for offloading of big data applications on cloud-edge platforms. *Future Internet*, 12(2), 2020.

[9] Y. Dong and Y.-D. Yao. IoT platform for COVID-19 prevention and control: A survey. *IEEE Access*, 9:49929–49941, 2021.

[10] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya. Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare. *Future Generation Computer Systems*, 78:659–676, 2018. Cited By :458.

[11] C. Feng, M. Adnan, A. Ahmad, A. Ullah, and H. U. Khan. Towards energy-efficient framework for IoT big data healthcare solutions. *Scientific Programming*, 2020, 2020.

[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[13] V. S. Gunge and P. S. Yalagi. Smart home automation: A literature review. *International Journal of Computer Applications*, 975(8887-8891), 2016.

[14] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, 47:98–115, 2015.

[15] M. S. Hossain and G. Muhammad. Emotion-aware connected healthcare big data towards 5G. *IEEE Internet of Things Journal*, 5(4):2399–2406, 2018.

[16] V. S. Kireev, S. A. Filippov, A. I. Guseva, P. V. Bochkaryov, I. A. Kuznetsov, V. Migalin, and S. S. Filin. Predictive repair and support of engineering systems based on distributed data processing model within an IoT concept. In *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 84–89. IEEE, 2018.

[17] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, Evidence-Based Software Engineering (EBSE) Project, Keele University and University of Durham, 2007. Version 2.3.

[18] D. Marikyan, S. Papagiannidis, and E. Alamanos. A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, 138:139–154, 2019.

[19] F. Mattern and C. Floerkemeier. From the Internet of Computers to the Internet of Things. In K. Sachs, I. Petrov, and P. Guerrero, editors, *From Active Data Management to Event-Based Systems and More*, volume 6462, pages 242–259. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[20] G. Mokhtari, A. Anvari-Moghaddam, and Q. Zhang. A new layered architecture for future big data-driven smart homes. *IEEE Access*, 7:19002–19012, 2019.

[21] L. Pani, A. Dutta, C. Misra, and R. Roy. Assisting fog-cloud computing with an adaptive traffic awareness resource provisioning algorithm for health data. In *2021 19th OITS International Conference on Information Technology (OCIT)*, pages 290–295, 2021.

[22] L. M. Pham, T.-T. Nguyen, and T.-Q. Hoang. Towards an elastic fog-computing framework for IoT big data analytics applications. *Wireless Communications and Mobile Computing*, 2021, 2021.

[23] G. Ramirez-Prado, B. Barmada, and V. Liesaputra. On non-intrusive prediction of activities and behavior. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6203–6204, 2019.

[24] R. Shehab, M. Taher, and H. K. Mohamed. Live big data analytics resource management techniques in fog computing for TeleHealth applications. *Jordanian Journal of Computers and Information Technology*, 7(1):89–103, 2021.

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

[26] S. Singh and A. Yassine. IoT big data analytics with fog computing for household energy management in smart grids. In A.-S. K. Pathan, Z. M. Fadlullah, and M. Guerroumi, editors, *Smart Grid and Internet of Things*, pages 13–22, Cham, 2019. Springer International Publishing.

[27] M. Weiser. The computer for the 21^{st} century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.

[28] J. Wu, P. Zhou, Q. Chen, Z. Xu, X. Ding, and J. Hao. Blockchain-based privacy-aware contextual online learning for collabrative edge-cloud-enabled nursing system in Internet of Things. *IEEE Internet of Things Journal*, 2021. Early Access.

[29] C. Xia, W. Li, X. Chang, F. C. Delicato, T. Yang, and A. Y. Zomaya. Edge-based energy management for smart homes. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pages 849–856, 2018.

[30] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu. Big data and cloud computing: Innovation opportunities and challenges. *International Journal of Digital Earth*, 10(1):13–53, 2017.

[31] H. Yar, A. S. Imran, Z. A. Khan, M. Sajjad, and Z. Kastrati. Towards smart home automation using IoT-enabled edge-computing paradigm. *Sensors*, 21(14):4932, 2021.

[32] A. Yassine, S. Singh, M. S. Hossain, and G. Muhammad. IoT big data analytics for smart homes with fog and cloud computing. *Future Generation Computer Systems*, 91:563–573, 2019.

Contributions

Coming up with a reserach question and scope was the work of the whole group. The search results were divided between the group members and independently reviewed and either included or excluded. Table 4 indicates who participated in writing a particular section of this document. Florian did try to fix most of the grammer and spelling mistakes, and to make the whole document consistent and coherent.

Table 4: Contribution

	Sections
Florian	Abstract, 1, 3, 5, 6, 7
Okke	2.4, 2.5, 4, 6
Anne-Ruth	Abstract, 2.1, 2.2, 2.3, 5

The State of the Art of Single Sign-On

Radu Geacu (2731957) **Bas Loyen (2754714)** **Mark Bebawy (2620527)**

Web Services and Cloud-Based Systems: Group 16

University of Amsterdam

Abstract

Single sign-on (SSO) is a mechanism that allows users to perform a single action of authentication, after which they can access a variety of services without being prompted to log in again. Many big companies such as Facebook, Google, and Microsoft are using SSO for their services extensively. There exist a variety of protocols that enable SSO, such as Web Services Federation (WS-Fed), SAML 2.0, OAuth 2.0, and OpenID Connect. The goal of this paper is to give an overview of the state-of-the-art of SSO by analyzing these protocols, discussing how secure they are, and researching to what extent they are used in industry. We do this by providing a literature review on papers that discuss (the security of) these protocols and on surveys on SSO. We conclude that OpenID Connect is currently the most secure and widely used protocol, and we give a prediction of the future of SSO and possible future work to be done in this field.

1 Introduction

Web services require their users to authenticate themselves before accessing protected resources. Historically, each service would have to come up with their own authentication mechanism, while users would have to replicate their identities [1]. *Single sign-on (SSO)* systems allow a user to perform a single action of authentication and then access a variety of services without being prompted to log in again. This increases their productivity and allows for more mobility between systems [2]. SSO is also user-friendly, as users only have to remember one set of credentials to access multiple services. The growth of companies such as Facebook, Google, Microsoft resulted in both the proliferation of the necessary technologies for SSO, and the concentration of many users in few places [2].

1.1 Research Questions

In this paper, we aim to analyze the evolution of SSO and, more specifically, the building blocks that enable it. To facilitate this analysis, our main research question is: *What is the state of the art of SSO?* To answer this question, we use the following sub-questions:

- What are the protocols/frameworks that enable SSO?
- How secure are these?
- What methods are currently used in industry for SSO?

1.2 Paper structure

We aim to offer a chronological overview of the evolution of SSO and its enabling frameworks, starting with *Section 2* where we discuss Web Services Federation. In *Section 3*, we analyze SAML 2.0.

Next, in *Section 4*, we evaluate OAuth 2.0; we recognize that technically OAuth is an authorization framework and it cannot constitute an SSO system by itself, but it is being adopted as a means of providing SSO [3] and it is a fundamental building block of OpenID Connect, which we discuss in *Section 5*. *Section 6* consists of a discussion of our findings and a comparison of the different methods we present, and we conclude by answering our research questions in *Section 7*.

2 Web Services Federation

2.1 Overview of WS-Trust

Before discussing Web Services Federation, a brief overview of WS-Trust and related mechanisms, which Web Services Federation builds upon, will be given [4]. These mechanisms are WS-Security, WS-Trust, and WS-SecurityPolicy, which provide a basic model for federation between providers and relying parties. WS-Security defines mechanisms to assure message integrity, authenticity and confidentiality. WS-SecurityPolicy enables the description of the security requirements of participating services, this includes the algorithms and types of tokens each service accepts. WS-Trust enables the use of the Security Token Service (STS) and a protocol for requesting or issuing the tokens. The STS tokens are used by WS-Security and described by WS-SecurityPolicy.

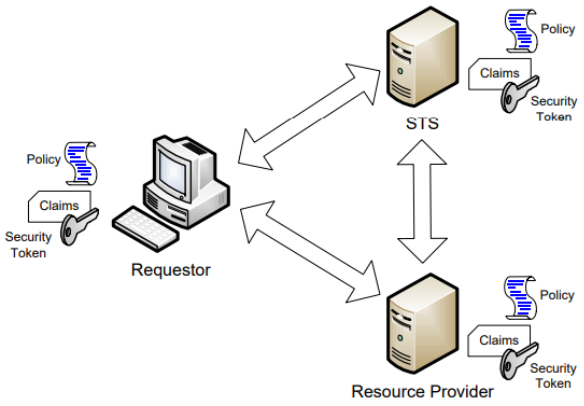


Figure 1: Security Token Service (STS) Model (from [4]).

In Figure 1 an overview of the STS model is given. Each arrow represents a potential path for communication. Each participant has their own policies which determine the security tokens and associated claims which are required to communicate with each participant. Whenever a requestor wants to gain access to certain resources, it queries the corresponding resource provider for its policies. Using policy expressions, the requestor checks whether it meets the security requirements of the resource provider. If the requestor's token does not meet the requirements described in the resource provider's policy, it might be able to request a token from another appropriate STS. The requestor can query this STS, which all have their own policy, to determine the requirements for requesting the type of token it needs.

WS-Trust defines independent protocol mechanisms for requesting, issuing, renewing, cancelling and validating security tokens. These security tokens can be exchanged to authenticate principals and protect resources. At the core of this protocol is the request-response message pair, Request Security Token (RST) and Request Security Token Response (RSTR). An example of both an RST and RSTR message are shown in Figure 2 and Figure 3, respectively.

2.2 Overview of WS-Federation

Web Services Federation ("WS-Federation" or "WS-Fed" for short) extends WS-Trust, and related mechanisms, as discussed in more detail in Section 2.1. WS-Federation introduces federations. A federation is a collection of realms (security domains) that have established relationships for securely sharing resources between each other [4].

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
    </wsa:Action>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <wst:RequestSecurityToken>
      <wst:TokenType>
        http://example.org/mySpecialToken
      </wst:TokenType>
      <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
      </wst:RequestType>
    </wst:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

Figure 2: Example of RST message (from [4]).

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Issue
    </wsa:Action>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse>
        <wst:RequestedSecurityToken>
          <xyz:CustomToken xmlns:xyz="...">
            ...
          </xyz:CustomToken>
        </wst:RequestedSecurityToken>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </s:Body>
</s:Envelope>
```

Figure 3: Example of RSTR message (from [4]).

To establish a federation with identity and recourse providers operating in different realms, they need to agree on what claims are required and frequently requires agreement on mechanisms to securely transport those claims between realms over possible unprotected networks. Generally, participants in a federation need to communicate these requirements over a wide variety of trust and communication topologies. To support the different topologies which the federations span, metadata needs to be exchanged. This metadata describes the endpoint references where services may be obtained and the potential security policies and communication requirements needed to access these endpoints. This exchange of metadata can become quite complicated, depending on the different policies of participants in a federation.

2.2.1 Components

When an organization establishes a federation, the participants need to exchange configuration information, this is the federation metadata. This will allow participants to identify the common services in the federation and retrieve the policies to access them. A mechanism for determining the authenticity must be in place. Services may also participate in multiple federations, so it must be possible to distinguish between these different contexts. Lastly, it is desirable for this configuration process to be automated. WS-Federation enables this by defining a metadata model and document format which extend the metadata already in place. This model describes how federation metadata about related services can be discovered/combined.

An authorization service may be implemented as a special type of Security Token Service to provide decision brokering services for participants. Extensions to the RST and RSTR mechanisms, as described in Section 2.1, can be sued to communicate authorization requests and decision outputs.

WS-Federation defines several authentication types in the form of Universal Resource Identifiers (URIs). These are used within the RST and RSTR messages to specify common authentication types and assurance levels. Since some of the data in these RST and RSTR messages can be quite sensitive, there are also options to indicate whether the messages should be encrypted.

2.3 WS-Federation in the real world

WS-Federation seemed to be mostly positively received and was covered quite a bit [5, 6, 7]. It was used by governments and other institutions, like CERN [8], all over the world. However, it is currently not used as much. This also becomes apparent when examining the years in which the covered literature was published. Most of the literature on WS-Federation was published before 2010 which already indicates that it is no longer state of the art.

3 SAML 2.0

3.1 Overview of SAML 2.0

Security Markup Language (SAML) is an XML-based protocol used for authorization and authentication between identity providers and resource providers [2, 9]. These XML-based messages are called Assertions. There are multiple types of Assertions. Authentication Assertions detail whether users are authenticated, Attribute Assertions detail what kind of rights, roles and access they have

and Authorization Assertions detail how they can use this data and resource. HTTP, SMTP, FTP and SOAP or other protocols can be used to transmit these assertions between security domains.

All of this gives SAML obvious applications in SSO. An example of the flow of SSO with SAML is shown in Figure 4. In this figure, we assume user U is already authenticated by source site S. The flow begins once user U returns to source site S. This could be after being redirected by destination site D for example. An assertion about the U's identity is stored by S if it recognizes U's browser B during the so-called user tracking. It then redirects the user's browser B to the destination site D. A small piece of data is included into the redirect by S, a SAML artifact. The SAML artifact refers to the stored assertion. Once D receives the redirect, it shows the artifact to S and requests the corresponding assertion. S confirms to D that U, presenting the artifact, was indeed authenticated by S.

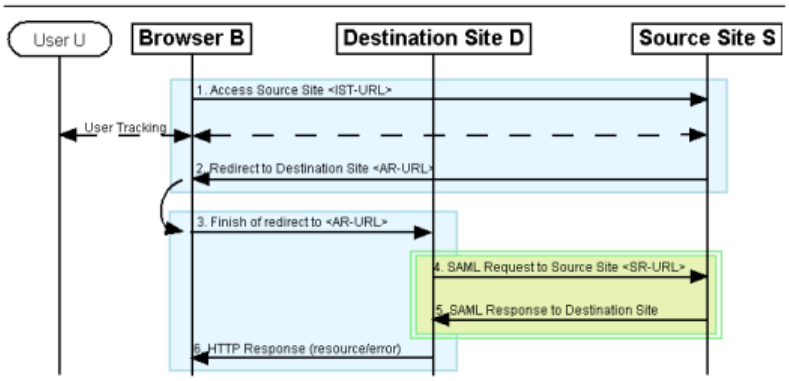


Figure 4: SAML SSO protocol flow (from [10]).

3.2 Security & Vulnerabilities

The security of SAML SSO heavily depends on several assumptions, such as the trust relationships between parties and the security mechanisms used to transport the messages. There are many security recommendations in the SAML specifications which help avoid certain pitfalls, but are definitely not an insurance policy for all security threats. This makes it quite difficult to achieve the needed level of security even for very simple SAML implementations. This can also lead to certain security threats in the real world such as Google's SAML SP-Initiated SSO implementation, where an attacker could take complete control over the authorization and authentication of hosted user accounts that can access web-based applications like Gmail or Google Calendar [11].

Further analysis of SAML reveals several flaws in the specification that can lead to vulnerable implementations [10]. These attacks include a replay attack, connection hijacking, HTTP referrer attack and a man-in-the-middle attack. These attacks could prove to be real threats to the security of an SSO implementation. Although it is generally considered to be a well-written protocol, it is not perfect and some changes are needed when implementing SSO using SAML.

3.3 SAML 2.0 in the real world

Just as with WS-Federation, a lot of the literature regarding SAML 2.0 is quite old (from before 2010). It is still being used, but as time has gone on newer and better protocols have taken SAML 2.0's place. It is still in use in some places, but as mentioned before, a lot of vulnerabilities have to be manually prevented by the developer.

4 OAuth 2.0

4.1 Overview of the OAuth 2.0 framework

The OAuth 2.0 ("Open Authorization") framework is a set of standards that enables a third-party application to obtain limited access to an HTTP service on behalf of a resource owner [12]. OAuth 2.0 (or *OAuth*[1] for short) allows a *resource owner* (user) to grant *relying parties* (RPs) or *clients* (web services) access to their resources (data) hosted on other services, called *identity providers* (IdP), *authorization servers*, or *resource servers*. [13] OAuth enables users to grant third-party clients access to their web resources without sharing login credentials or their complete data. [14]

Initially, OAuth's objective was to serve the *authorization* needs for websites (that is, to grant the right permissions to authenticated users). However, as it became widely adopted across industry, it has been re-purposed to also serve *authentication*[2] purposes (that is, to verify the identity of users). Major providers such as Facebook, Google, and Microsoft support this scheme [13]. More than that, it has also been targeted towards mobile platforms [15]. Recently, its use in IoT applications is also under review [16, 17, 18].

It can therefore also be used as a single sign-on (SSO) scheme: the relying party (i.e., the application that requires authentication) relies on the identity provider (for example, Facebook) to authenticate the user. This is based on browser redirection: the relying party redirects the user's browser to an identity provider; the IdP handles authentication, asks the user for permission to grant access to (some) resources to the RP, and then redirects the user back to the RP web service (with an access token); the RP can use this access token to call the IdP's APIs and access the user's profile [14].

As early as 2012, OAuth became the most widely supported protocol for API authorization, largely due to its adoption by giants such as Google, Twitter, Facebook [19]. Besides these, it is also used by identity providers such as Amazon, Microsoft, LinkedIn, Yahoo, and GitHub, making OAuth one of the most used single sign-on systems in the world [13].

4.1.1 Predecessor

OAuth 1.0, published in 2009, was intended to unify existing authorization mechanisms implemented by different services, such as Twitter, Google, and Flick. However, it was criticized as being "too complex, inflexible, and website-centric" [19]. OAuth 2.0, published in 2012, is the latest major revision to this protocol (and it is not backwards compatible with OAuth 1.0) designed to reduce client developer complexity (which drew criticism for sacrificing security for usability [20]; for example, unlike OAuth 1.0, the new framework does not feature *request and response authentication* [19]).

4.1.2 Components

The OAuth framework is built on several central components, starting with *roles*. Four roles are defined [12]: resource owner (an entity capable of granting access to a resource; for example, the end-user of a Facebook account), resource server (server hosting the protected resources, uses access tokens to respond to requests), client (an application making a protected resource request), and authorization server (the server that issues access tokens to the client after authenticating the resource owner and obtaining their authorization).

Scopes are bundles of permissions asked for by the client when requesting a token.

Four common *authorization types* (also called *modes*) are defined and analyzed across the literature [13, 20].

- *Authorization code*. When a user wants to authorize an RP to access their data, they are first redirected to the IdP where they must authenticate. The user is redirected back to the RP with an authorization code generated by the IdP which is used further by the RP (along client secrets) to obtain an access token, which itself is used as a credential to access protected resources hosted by the IdP. This mode is also known as the "server-flow" [14] and can be seen in Figure 5.

[1] We use OAuth as shorthand for OAuth 2.0 unless otherwise mentioned

[2] Some literature distinguishes this mechanism as *pseudo-authentication*

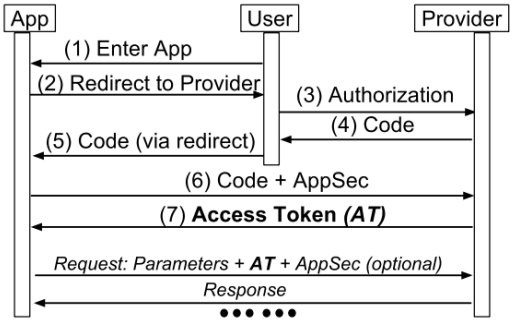


Figure 5: OAuth 2.0 flow in case of authorization-code mode (from [21]).

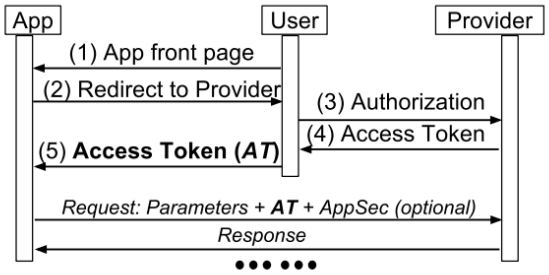


Figure 6: OAuth 2.0 flow in case of implicit mode (from [21]).

- *Implicit mode*. The resource owner's authorization is given directly as an access token, thereby skipping the authorization code steps. This mode is also known as the "client-flow" [14] and can be seen in Figure 6.
- *Resource owner password credentials*. The user gives their credentials for the IdP directly to the RP. The RP can then authenticate on the user's behalf. This mode is intended for highly trusted RPs.
- *Client credentials*. The client and the resource owner are considered the same entity, and the user's interaction is not required. The RP accesses its own resources at the IdP.

Analyzing the security of OAuth is a "challenging task" due to its various features and the "inherent complexity of the web" [13]. Most analysis efforts have focused on specific implementations [21, 15, 22, 23], not on the standard itself. Moreover, many existing approaches analyze only the authorization and implicit modes of OAuth, not all modes running simultaneously [13]. Therefore, we distinguish two types of studies in the literature: those that focus on a formal model of OAuth and those that focus on empirical real-world implementations.

4.2 Formal analysis

One of the first formal verification of security of the OAuth standard is done in [20]. The standard is formalized using the Alloy framework according to guidelines defined by the authors. They formally confirm a security flaw found in previous research (which shows that, in cases where the client is a desktop application with no server-side security measures, client passwords are vulnerable). However, the paper is meant only to present the Alloy formalization as a proof-of-concept, so no further flaws are presented and are left for future work.

The authors of [19] perform a comprehensive analysis of OAuth using formal methods, with models built using the WebSpi library and verification done using ProVerif. While relying on general assumptions (network attackers, bad-acting resource owners/clients, untrusted websites, malicious

JavaScript are enabled, but every authorization server is honest and no CSRF attacks[3] exist on honest applications) the authors are unable to find any attacks. If these assumptions are changed, attack possibilities are found and, using these, the authors in fact find vulnerabilities in implementations of different web services, such as Facebook, Twitter, Yahoo, IMDB. The authors also find that in certain conditions CSRF attacks are possible (and exist in real-world implementations), as well as token redirection attacks (which rely on the existence of open redirections on the client).

More recently, the authors of [13] perform the first extensive formal analysis of the OAuth standard for all four modes. They use a pre-defined "comprehensive" formal model of the web [24] and incorporate best practices in terms of security to model OAuth with as few assumptions as possible. Three security properties are formalized: authorization, authentication and session integrity. During analysis, four attacks on OAuth were in fact identified: identity providers unintentionally forwarded user credentials to clients or a network attacker can impersonate any identity provider (breaking the first two properties); an attacker can force browsers to be logged in under the attacker's name to the identity provider (breaking session integrity). After notifying the working groups of the affected frameworks, and modelling OAuth with the fixes in place, the authors prove that it does finally satisfy the security properties. They emphasize that not only do the aforementioned attacks have to be mitigated, but also that standard best security practices (such as the ones found in [25]) are essential in individual implementations, where the largest security issues can be found. Their work differs from (and improves on) other research through the "more expressive" model that is used and the fact that OAuth as a standard is analyzed, as opposed to specific implementations.

4.3 Empirical analysis

Shortly after the protocol was published, the authors of [14] studied OAuth implementations of three major identity providers (Facebook, Microsoft, Google) and 96 relying parties. They find that, despite the fact that OAuth was designed to prevent exposing access tokens over a network, many tokens obtained are transmitted in unprotected form to the relying party. For instance, tokens are appended as query parameters to the URI of the relying party's sign-in endpoint, or stored as cookies. Moreover, the authors found that only 21% of relying parties employed SSL to protect sessions at the time of writing. The authors also find that tokens can be stolen via malicious JavaScript, through impersonation, session swapping or CSRF attacks. The authors assert that these vulnerabilities are mainly caused by design decisions of OAuth that trade security for simplicity, and that leave security decisions at the hands of clients or identity providers.

These findings are also corroborated in [23]. The authors identify the following attack methods: replay attack module (an attacker may capture an authorization code redirect request while the client communicates with the resource owner's agent); phishing attacks (client applications may be illegitimate, and even though they are allowed to consume services from, for example, Google, they might not have an interest in user privacy); impersonation attacks (as in [14], the authors find vulnerabilities caused by the fact that the client application is not required to use any TLS mechanism to protect the data).

The authors of [22] focus on specific implementations of OAuth in China: 60 relying parties that support SSO via identity federation are analyzed. The authors identify "serious" vulnerabilities in a number of these systems, and make recommendations based on that. First, countermeasures against CSRF attacks are needed (relying parties in this study do not implement them, despite recommendations from identity providers to include the *state* parameter in the requests). Next, relying parties must ensure that a non-guessable parameter is generated, and that *state* is session dependent.

Similar results can be found in a study of the 50 most popular websites in Ireland, which found that of the 21 that had OAuth support, 2 were susceptible to some form of attack in the threat model defined in [25, 26]. Besides these findings, the authors implement their own OAuth environment: a client application, a resource server and an authorization server, and test its robustness against the threat model. They find that their solution performs well against threats because it follows guidance and is "implemented correctly".

[3] CSRF (cross-site forgery requests) is a type of exploit where unauthorized commands are issued from a user that the application trusts by forwarding their session cookie. A common countermeasure against CSRF is to require a session-specific nonce that is difficult to be forged to be sent with every request [19].

While most other studies rely on manual empirical analysis to discover new vulnerabilities in OAuth implementations, or only perform automated testing for a set of specific and previously-known vulnerabilities, the authors of [27] propose an adaptive model-based testing tool to enable the automated discovery of vulnerabilities. The authors confirm rediscovering existing vulnerabilities and, also, they claim the discovery of new vulnerabilities: misuse of the *state* parameter, amplification attacks via dual-role identifying parties (those that serve both as an IdP and as a client to another IdP), failure to revoke authorization, and failure to adopt TLS protection. However, all of their findings are, again, related to the actual implementation of OAuth protocols, and not to the standard itself.

It is not, however, always the case that implementation flaws are the main vulnerability of OAuth. The authors of [21] argue that user data can leak from various points due to fundamental design decisions of the protocol: "OAuth focuses on protecting the user, not the application". They define this as the "app impersonation attack". First, if a provider supports the implicit flow of OAuth, a user can bypass the client and directly query the resource provider's API. Second, depending on the type of access token that is issued by the IdP (bearer token[4] or MAC token[5]), the token can be forged. The authors also conduct a real-world study of 12 major providers and found that 8 of them present these vulnerabilities (in one case, there was the possibility of a leak of 200 million users' data because a read token could be used to access any objects). To mitigate such vulnerabilities, the authors propose that providers opt-out of certain OAuth modes and review the access scopes in their architecture (we would argue, however, that at least the second proposal has nothing to do with OAuth as a standard, and more to do with a provider's implementation and, possibly bad, decisions). It is also worth mentioning that the implicit mode has been discontinued and replaced by new guidance [28].

Widespread adoption of OAuth for mobile platforms has triggered research interest into its vulnerabilities in this area [15, 29]. For example, in [15] the authors find that 59.7 % of the 149 applications surveyed were not implemented correctly and therefore vulnerable. More specifically, the difference between redirection mechanisms on browsers vs mobile platforms (custom scheme mechanism on iOS and Intents on Android), and the fact that the OAuth specification "makes intensive use of HTTP redirections" leads to flaws in implementations. Moreover, in web applications the browser is entrusted to deliver a message (confidential messages such as access tokens) to the intended target; the authors find that for mobile application many developers have "ill-conceived notions of the real recipients of messages", which means that it is difficult for a message provider to *ensure* that a token is sent to the intended recipient. In [29] the authors corroborate these findings: 58.7% of Google Play Android applications present flawed implementations of the OAuth protocol. Vulnerabilities identified include: improper user-agent (WebView and System browser are "not suitable" for authentication of a relying party's app), lack of authentication (Intents can be used by a malicious app to hijack data sent by a legitimate relying party app), incorrect implementations of TLS, insecure secret management and problematic server-side validation.

4.4 Vulnerabilities summarized

Broad vulnerabilities identified in OAuth can be grouped as follows:

- **Access token eavesdropping [14, 23, 27].** An attacker can sniff the unencrypted communication between a browser and a client application/relying party [14]. Failure to adopt TLS protection increases this vulnerability [27, 29].
- **CSRF attacks (cross-request site forgery) [14, 22, 26, 27].** A user can be tricked into loading a page that contains a malicious request that can affect the integrity of the user's session data [14]. The OAuth specification recommends including the state parameter to protect against this sort of attacks; the relying party can then verify the source of a request [22]. It is also worth mentioning that the RFC for OAuth states that both the client and the authorization server "must" implement CSRF protection [12]. However, as [30] found, only 4 out of 11 major IdPs require CSRF defenses, while the reset either suggest it or ignore it.
- **Impersonation [14, 22, 23, 27].** An attacker can send a stolen or guessed SSO credential to the RP's endpoint. To mitigate this, the SSO credential must be limited to one-time use

[4]Bearer token: simply added to the HTTP request header and any party in its possession can get access to a resource without verifying their identity

[5]MAC token: a client secret is used to compute a cryptographically secure version of the key, thereby ensuring proof of identity when making a request

[27] or the RP can check if the response sent is by the same browser which performed the authorization request [14]. This is also called a "replay attack module" [23]. However, protection against this sort of attacks is left at the hands of the RP and whether they decide to mitigate it or not [22].

- **Session swapping.** If the RP does not provide a state parameter in an authorization request, then an attacker can exploit this contextual binding vulnerability [14].
- **Access token theft.** The authors of [14] distinguish the case when a token is stolen via XSS[6]: malicious JavaScript can be used to exploit the fact that, in some cases, an IdP might perform automatic authorization granting (if permissions related to the request have been granted by the user previously and if the user is logged in to the IdP from the browser in the same session). The authors of [23] describe a phishing method, in which an attacker can poison DNS records to ensure that a user that tries to visit legitimate sites is redirected to malicious ones.

5 OpenID Connect

5.1 Overview of the OpenID Connect framework

In this section we will discuss the OpenID Connect (OIDC) protocol for single sign-on (SSO) that was standardized in 2014 [31]. At the moment of writing, OIDC is the most widely used protocol for SSO [32] that has been adopted by large companies like Facebook, Google, Twitter, LinkedIn, Amazon, Microsoft, and Salesforce [33]. OIDC is based on the OAuth 2.0 protocol that we discussed in the previous section and is used to enable clients to verify the identity of users based on authentication that is performed by an authorization server. As discussed in the previous section, OAuth 2.0 provides a framework to obtain access to HTTP resources and was originally designed as an authorization framework. However, unlike OIDC, OAuth 2.0 does not provide a standard for obtaining identity information [31]. We will discuss below how OIDC provides identity information.

A typical OIDC flow, as described in [34], is shown in Figure 7 and works as follows. A user browses to a *relying party (RP)*. The RP can be some application that implements SSO using OIDC, so that users can log in with their credentials from some *identity provider (IdP)*[7]. The user clicks a button that allows them to log in with SSO (often RPs have these buttons with texts like *Login with your Google account*). The RP then *discovers* some information about the IdP, namely the *authorization endpoint* (the IdP URL at which the user can fill in their credentials), the *redirect endpoint(s)* (one or more URLs to which the browser is redirectd by the IdP after authenticating themselves), the *token endpoint* (a URL from which the RP can obtain the ID token), the issuer identifier, the IdP's public key to verify the signature of the ID token, the *client id* (which the IdP uses to identify the RP), and possibly a *client secret* which the RP uses to prove its identity to the token endpoint. After discovery, the RP *registers* itself at the IdP (establishing a trust relationship between the RP and IdP by exchanging the RPs redirect URIs in exchange for a client id and client secret that the IdP issues to the RP). Then, the RP redirects the user to the IdP, where they log in using their (for example Google) credentials (or using some session cookie if they are already signed in to the IdP). If successful, the IdP provides a cryptographically signed *ID token* to the RP. The RP uses this token to verify the user's identity, after which the RP sets a session cookie in the browser, which indicates that the user is authenticated and signed in to the RP.

5.2 Components

5.2.1 ID token

The ID token is the primary extension of OIDC compared to OAuth 2.0 [31]. It is a JSON Web Token (JWT) that contains information (called *claims*) about the user and about the identity provider. There are a couple of required claims that this token should have [31]. They are:

[6] XSS: cross-site scripting, meaning that malicious scripts are injected into otherwise trustworthy websites

[7] In literature on OIDC, the relying party (RP) is sometimes called the *client* and the identity provider (IdP) is sometimes called the *OpenID provider (OP)*. To avoid confusion and for consistency with the section on OAuth 2.0, we will (like [34]) use the terms RP and IdP.

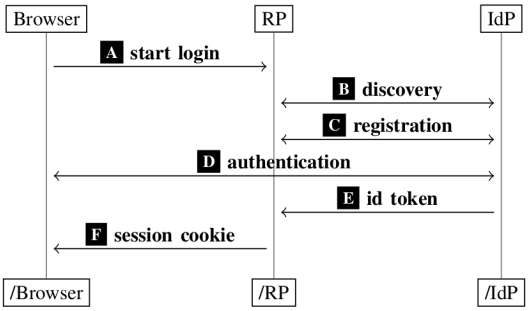


Figure 7: High level overview of OpenID Connect flow (from [34]).

- *iss.* This is an HTTPS URL (without query components) that identifies the *issuer* (i.e., the IdP).
- *sub.* This is a string that identifies the subject (i.e., the user) and is unique at the IdP.
- *aud.* This is a string (or array of strings) that represents the audience of the ID token. It contains (at least) the OAuth 2.0 *client id* of the RP.
- *exp.* Expiration time of the ID token.
- *iat.* Time at which ID token was issued.
- *nonce (required if the authentication request contains a nonce parameter).* The same nonce parameter as in the authentication request, used to mitigate replay attacks (see below).

5.2.2 Modes

As is the case with OAuth 2.0, OIDC defines the *authorization code mode* and the *implicit mode*. Additionally, OIDC also defines the *hybrid mode*. These modes define how the RP obtains the ID token from the IdP. The mode that is used is passed as part of the browser's request to the IdP before providing credentials [34].

- In the *authorization code mode*, the RP obtains the ID token from the IdP via direct communication. After the user authenticates their selves at the IdP, the IdP grants them an *authorization code* in the body of an HTTP redirect, which is used to redirect the browser to the RP. The RP receives the authorization code, presents it to the IdP at the token endpoint (together with the client id, client secret and redirect URI that was used to obtain the authorization code), after which the IdP issues an access token[8] and an ID token to the RP. The user is then logged in, and the RP can issue a session cookie to the browser.
- In the *implicit mode*, the RP obtains the ID token from the IdP via the browser. The flow is similar to the authorization code mode, but the IdP issues an ID token to the browser directly instead of issuing an authorization code. The RP then retrieves the ID token from the browser after the browser is redirected to the RP.
- In the *hybrid mode*, the RP can obtain the ID token from the IdP using direct communication and indirect communication via the browser at the same time. The browser receives an authorization code and either an ID token, an access token, or both (depending on the configuration of the IdP) when it gets redirected to the RP. The RP retrieves this information as done in the implicit mode, but additionally the RP can use the authorization code to retrieve a (second) ID token and access token from the IdP.

5.3 Security analysis

As OpenID Connect is a newer protocol than OAuth 2.0, less research has been done on its security. This is also due to the fact that OpenID Connect still uses a major component of OAuth 2.0, which means that some of the attacks on OAuth 2.0 can also be done on OpenID Connect. Below, we will

[8]The access token is used for delegated authorization as explained in Section 4.

compare the security of these protocols and discuss how they relate. We start, however, by focusing on the research that has been done on the security of OpenID Connect in chronological order.

In 2016, the authors of [35] did a practical analysis of the security of Google's implementation of OpenID Connect. To this end, they went through all the GTMetrix Top 1000 Sites[9] and selected the sites that provide services in English and support authentication with Google's OpenID Connect service. They found 103 such sites for which they used Fiddler[10] to analyze the HTTP traffic between the RPs (the websites) and the IdP (Google's OpenID Connect service). They found some severe vulnerabilities in many of these sites that allow an attacker to impersonate a user and log in to the RP. They verified these attacks by creating accounts for these purposes and successfully executing the attacks on these accounts. They concluded that these vulnerabilities originate from Google's design of their OpenID Connect service and from design choices by the RPs who (unknowingly) traded off security for ease of implementation. They also give practical recommendations on how to mitigate these vulnerabilities and on how to implement OpenID Connect securely. This is the same conclusion that we found for OAuth 2.0: the OIDC protocol is secure *if implemented correctly*. It would have been interesting if the authors also studied the vulnerabilities of less popular websites that use Google's OIDC service, as these websites are developed by smaller (maybe less experienced) teams, which would be a great indication of how manageable it is for small teams or individual developers to implement OpenID Connect correctly and securely. Also noteworthy is the fact that the researchers treated the RP and IdP as black-boxes and based their study solely on the HTTP traffic between these parties. This means that there may be uncovered vulnerabilities and implementation flaws for which future research is necessary.

In 2017, the authors of [36] studied the security of OpenID Connect by applying well-known SSO attacks to it. Where necessary, they modified the attacks to make them suitable with OIDC. Surprisingly, they found that at that time, 75% of the officially referenced OIDC libraries were vulnerable to some of these attacks. They distinguished *single-phase attacks* (which abuse a lack of a single security check) from *cross-phase attacks* (which require a complex setup to modify various messages throughout the entire protocol flow). A great contribution of this paper is an open-source Evaluation-as-a-Service tool[11] that automatically evaluates the security of an OpenID Connect implementation, finds vulnerabilities, and gives recommendations on how to mitigate them. The single-phase attacks identified by this paper result from wrong implementations of OpenID Connect, as the specification mentions these attacks and explains the necessary verification steps that are needed to avoid these attacks. We discuss two of the four single phase attacks discussed in this paper (we do not discuss all of them, as all attacks can be mitigated by implementing the OIDC specification correctly; the cross-phase attacks are more interesting for us).

- **Wrong Recipient.** Suppose a malicious RP (mRP) receives an ID token of user A who just signed in to mRP. The mRP can then use this token to try to impersonate user A by sending the token to another RP (RP2) that implements OIDC. If RP2 does not verify the *Recipient* information in the ID token (that is, the *aud* claim which indicates that the audience of this token is mRP), then mRP gets access to user A's resources in RP2. If, however, RP2 correctly verifies the *aud* claim (checking that the token was issued by the IdP to RP2), then it will reject mRP's token and this attack is not possible.
- **Replay attack.** As is the case with OAuth 2.0, OIDC is also vulnerable to replay attacks. A replay attack can occur if an RP does not verify the *exp, iat,* or *nonce* claims in the ID token. These claims indicate the freshness of the token. If, for example, the *exp* claim is not verified, then an ex-employee can use their once obtained ID token to keep signing in to their old company's RP as the expiration of the token is never checked. The *nonce* claim is also used to mitigate this attack; the value of this claim in the token should be equal to its value in the authentication request, binding the token to that specific request. Of course, the attack is only mitigated if these claims are verified by the RP.

The paper mentions three cross-phase attacks that allowed an attacker to skip a verification step by ingeniously abusing the lack of a binding between different phases of the OIDC protocol. One of them (*Issuer Confusion*) is an implementation flaw, which we omit from this paper, as it can be mitigated by implementing OIDC correctly. The other two attacks (*IdP Confusion* and *Malicious*

[9] `https://gtmetrix.com/top1000.html`

[10] `https://www.telerik.com/fiddler`

[11] `https://github.com/RUB-NDS/PrOfESSOS`

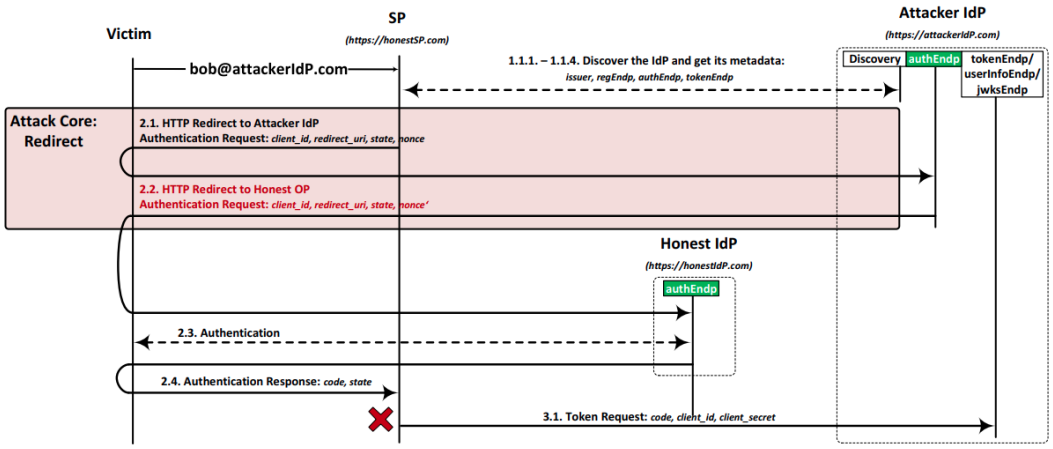


Figure 8: OpenID Connect vulnerability: IdP Confusion attack flow; figure from [36], where the authors use the term *Service Provider (SP)* for the object that we call *Relying Party (RP)*.

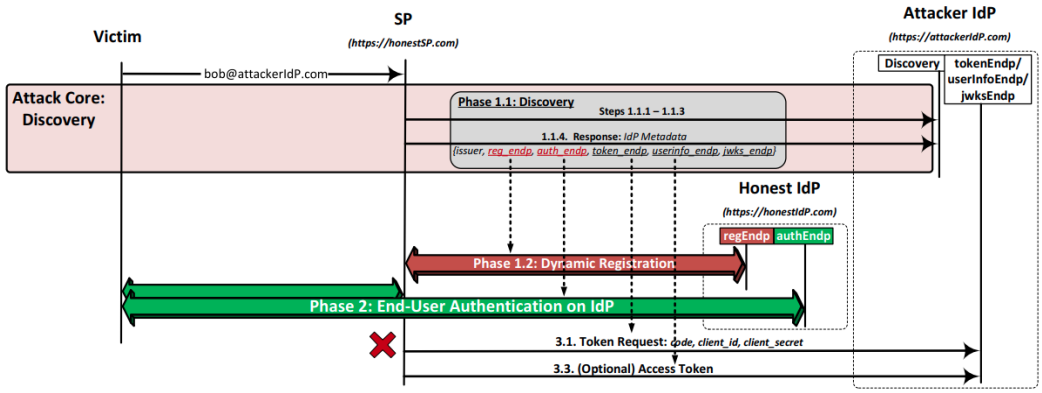


Figure 9: OpenID Connect vulnerability: Malicious endpoints attack flow; figure from [36], where the authors use the term *Service Provider (SP)* for the object that we call *Relying Party (RP)*.

Endpoints) exposed a specification flaw that resulted in an updated OIDC specification. By virtue of this study, the current OIDC protocol incorporates some changes that mitigate these attacks (we discuss them below). Both cross-phase attacks involve the discovery phase of the protocol where, as discussed in Subsection 5.1, various endpoint URLs are exchanged between the IdP and the RP.

- **IdP Confusion attack.** For this attack, the attacker implements a custom IdP to steal an authorization code from an honest IdP. The attacker can then get access to an honest RP. This attack assumes that the RP has the same *client id* for both the attacker IdP and the honest IdP (the OIDC specification allows this). The attack is visualized in Figure 8. First, a user clicks on a malicious link that initiates the OIDC flow for the honest RP and the attacker IdP. In steps 1.1.1 to 1.1.4, the RP discovers endpoints from the attacker IdP. Then, in step 2.1 the user is redirected to the attacker IdP, which in turn redirects the user to the honest IdP in step 2.2. In step 2.3 the user gets asked to fill in their credentials to authenticate to the honest IdP (this is the only step where the user has a chance to not fall into this attack by not authenticating). If the authentication is successful, the honest IdP sends an authorization code to the RP in step 2.4. The RP, however, still thinks that the user is authenticating to the attacker IdP, so it presents the authorization code to the attacker IdP to obtain the ID token. The attacker now has an authorization code which they can use to log in to the RP, impersonating the victim user (as the honest IdP issued the authorization code and will upon reception exchange it for an ID token). This attack was possible, because the user authentication step was not linked to the ID token request step in the protocol. In the current

version of the protocol, this is mitigated by adding an *issuer* parameter to every step in the protocol which indicates who the IdP is that issues the tokens.

- **Malicious Endpoints attack.** The malicious endpoints attack is very similar to the IdP Confusion attack, but here the attacker abuses the fact that the discovery step was not linked to the ID token request step (whereas the IdP Confusion attack abuses the fact that the authentication step was not linked to the ID token request step). Again, the user clicks on a malicious link that initiates the OIDC flow for the honest RP and the attacker IdP. In phase 1.1, the RP discovers endpoints from the attacker IdP, but (and this is the major difference between these two attacks), here the attacker IdP returns malicious endpoints. More specifically, the issuer, token endpoint, and user information endpoint are set to the attacker IdP, but the registration and authentication endpoints are set to the honest IdP. This results in phases 1.2 and 2, where the RP registers at the honest IdP and the user authenticates to the honest IdP. The resulting authorization code (and client id and client secret) are then sent to the (malicious) token endpoint, which points to the attacker IdP. Again, the attacker obtained an authorization code which they can use to log in to the RP, impersonating the victim user. This attack can also be mitigated by connecting all steps of the protocol through an *issuer* parameter.

In 2017, the first formal security analysis of OpenID Connect was performed [34]. In this paper, the authors give a long list of possible attacks on OIDC (including the attacks from [36] which we discussed above) with implementation choices that mitigate these attacks. They use a formal model of the web to prove that OIDC is secure when following these implementation choices. We omit the details on the attacker models in this analysis, as we are only interested in practical security of OIDC to determine to what extent it is the state-of-the-art of SSO. The paper concludes that OIDC is highly secure regarding authentication, authorization, and session integrity *if users implement it correctly*.

Besides security, there are also some privacy concerns regarding OIDC. For example, the IdP learns which RPs users sign in to [32]. Studies like [32] and [37] propose extended and modified protocols that mitigate these privacy problems.

5.3.1 Relation to OAuth 2.0

The security of OIDC is similar to that of OAuth 2.0, in that it is secure if implemented correctly, and thus the security of the protocol depends on its implementation. The studies we reviewed show that although the OIDC specification is sound, developers find it hard to implement it correctly, resulting in unsafe usage of the protocol [33, 35, 36]. Furthermore, the discovery and registration steps of OIDC do not appear in OAuth 2.0 and thus they open the door to new attacks, as we saw above. OIDC is, however, less prone to some attacks of OAuth 2.0 [34] as OIDC requires some cryptographic signing steps which OAuth 2.0 does not require. Also, the nonce parameter that mitigates replay attacks in OIDC is not present in OAuth 2.0.

6 Discussion and Future Work

We discussed four of the most-used protocols for SSO, namely Web Services Federation (WS-Fed), SAML 2.0, OAuth 2.0, and OpenID Connect (OIDC). We found that although WS-Fed was well-received when it was published, it has in the meantime become outdated. SAML 2.0 is still being used by some enterprise applications, but OAuth 2.0 and OIDC are by far the most popular protocols today, with OIDC being most-widely used for SSO.

In the case of OAuth, though early formal security validations of the standard did find some vulnerabilities (although the authors admit, however, that "correct implementations" of OAuth 2.0 do not suffer from some attacks [19]), later research [13] formally proves that, if best security practices are followed, OAuth provides "strong authorization, authentication and session integrity". These best security practices are taken into account in their modeling and were defined in [25] (which, it is worth mentioning, is built based on some recommendations from these papers [13]). RFC6819 presents a threat model of the framework, grouped by the component that might be attacked: clients, authorization endpoints, token endpoints, authorization granting flows. Recommendation such as token encryption (and lifetime constraints and refresh times) or usage of the "state" parameter are already present in the OAuth standard (which is also constantly being updated with recent security findings [28]). This leads to what seems to be the largest security vulnerability of the OAuth standard:

most of the OAuth exploits can happen because of specific implementations, and not necessarily issues with the framework itself [26]. This is also related to the main criticism aimed at OAuth security that is common in the literature: that too much leeway, in terms of security, is left to clients (RPs) themselves and their implementation decisions. One possible recommendation, endorsed by [30], is to not trust developers to implement such protection against vulnerabilities, and to have identity providers enforce this before allowing clients to connect to their data.

We found recurring patterns when analyzing the security of the protocols, namely that the protocols were secure when implemented correctly. Sometimes, the protocol specifications were too complex, resulting in unsafe implementations of the protocols. As a result, the protocol specifications were modified or the protocol got changed (for example, adding the *issuer* parameter to every step of OIDC mitigated some protocol specification flaws). This convinces us that this pattern will continue in the future. OIDC still has security threats that cannot be mitigated if the specifications are not implemented correctly. We expect future work to be done to develop extensions of OIDC (or new protocols) that can only be implemented safely. Studies like [32, 37] are making the first steps towards this direction.

7 Conclusion

Recall that our main research question was: *What is the state of the art of SSO?* We answer this question by answering the three sub-questions that we stated in Section 1.1:

- What are the protocols/frameworks that enable SSO?
- How secure are these?
- What methods are currently used in industry for SSO?

We analyzed four main protocols that enable SSO, namely WS-Fed, SAML 2.0, OAuth 2.0, and OpenID Connect. All four methods are still being used in industry, although WS-Fed and SAML 2.0 are starting to get outdated, and OAuth 2.0's main purpose is authorization, not authentication. We found that all protocols are in principle secure if implemented correctly, although this requires quite a lot of effort in case of WS-Federation and SAML 2.0. Studies like [33, 34, 36] have shown that currently the protocols are not always implemented correctly, so there is still work that needs to be done (and is being done [32, 37]) to further secure the SSO protocols and to write more clear developer specifications for it. In conclusion, OpenID Connect constitutes the state-of-the-art of SSO, being the most-widely used protocol that is secure if implemented correctly, but further work is needed to mitigate the possibility to implement it wrongly, exposing RPs to security vulnerabilities.

References

[1] Victoria Beltran. "Characterization of web single sign-on protocols". In: *IEEE Communications Magazine* 54 (July 2016), pp. 24–30. DOI: 10.1109/MCOM.2016.7514160.

[2] V. Radha and D. Hitha Reddy. "A Survey on Single Sign-On Techniques". In: *Procedia Technology* 4 (2012). 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012, pp. 134–139. ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2012.05.019. URL: https://www.sciencedirect.com/science/article/pii/S2212017312002988.

[3] Wanpeng Li, Chris J Mitchell, and Thomas Chen. *OAuthGuard: Protecting User Security and Privacy with OAuth 2.0 and OpenID Connect*. 2019. DOI: 10.48550/ARXIV.1901.08960. URL: https://arxiv.org/abs/1901.08960.

[4] Marc Goodner et al. "Understanding ws-federation". In: *Microsoft and IBM* (2007).

[5] Michiaki Tatsubori, Takeshi Imamura, and Yuhichi Nakamura. "Best-practice patterns and tool support for configuring secure web services messaging". In: *Proceedings. IEEE International Conference on Web Services, 2004.* IEEE. 2004, pp. 244–251.

[6] Yuichi Nakamura et al. "Model-driven security based on a web services security architecture". In: *2005 IEEE International Conference on Services Computing (SCC'05) Vol-1*. Vol. 1. IEEE. 2005, pp. 7–15.

[7] Nils Agne Nordbotten. "XML and web services security standards". In: *IEEE Communications Surveys & Tutorials* 11.3 (2009), pp. 4–21.

[8] Emmanuel Ormancey. "CERN single sign on solution". In: *Journal of Physics: Conference Series*. Vol. 119. 8. IOP Publishing. 2008, p. 082008.

[9] Andreas Pashalidis and Chris J Mitchell. "A taxonomy of single sign-on systems". In: *Australasian conference on information security and privacy*. Springer. 2003, pp. 249–264.

[10] Thomas Groß. "Security analysis of the SAML single sign-on browser/artifact profile". In: *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE. 2003, pp. 298–307.

[11] Alessandro Armando et al. "Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps". In: *Proceedings of the 6th ACM workshop on Formal methods in security engineering*. 2008, pp. 1–10.

[12] D. Hardt. "RFC6749 - The OAuth 2.0 Authorization Framework". In: *IETF* (2012). URL: https://tools.ietf.org/html/rfc6749.

[13] Daniel Fett, Ralf Küsters, and Guido Schmitz. "A Comprehensive Formal Security Analysis of OAuth 2.0". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016).

[14] San-Tsai Sun and Konstantin Beznosov. "The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems". In: Oct. 2012, pp. 378–390. DOI: 10.1145/2382196.2382238.

[15] Eric Chen et al. "OAuth Demystified for Mobile Application Developers". In: Nov. 2014. DOI: 10.1145/2660267.2660323.

[16] Sowmya Ravidas et al. "Access control in Internet-of-Things: A survey". In: *Journal of Network and Computer Applications* (July 2019). DOI: 10.1016/j.jnca.2019.06.017.

[17] W. Denniss. "DRAFT - OAuth 2.0 Device Flow for Browserless and Input Constrained Devices". In: *IETF* (2017). URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-device-flow/09/.

[18] Shamini Emerson et al. "An OAuth based authentication mechanism for IoT networks". In: Oct. 2015, pp. 1072–1074. DOI: 10.1109/ICTC.2015.7354740.

[19] Chetan Bansal, Karthikeyan Bhargavan, and Sergio Maffeis. "Discovering Concrete Attacks on Website Authorization by Formal Analysis". In: *2012 IEEE 25th Computer Security Foundations Symposium*. 2012, pp. 247–262. DOI: 10.1109/CSF.2012.27.

[20] Suhas Pai et al. "Formal Verification of OAuth 2.0 Using Alloy Framework". In: June 2011. DOI: 10.1109/CSNT.2011.141.

[21] Pili Hu et al. "Application impersonation: Problems of OAuth and API design in online social networks". In: *COSN 2014 - Proceedings of the 2014 ACM Conference on Online Social Networks* (Oct. 2014), pp. 271–277. DOI: 10.1145/2660460.2660463.

[22] Wanpeng Li and Chris J. Mitchell. "Security Issues in OAuth 2.0 SSO Implementations". In: *Information Security*. Ed. by Sherman S. M. Chow et al. Cham: Springer International Publishing, 2014, pp. 529–541. ISBN: 978-3-319-13257-0.

[23] Feng Yang and Sathiamoorthy Manoharan. "A security analysis of the OAuth protocol". In: *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. 2013, pp. 271–276. DOI: 10.1109/PACRIM.2013.6625487.

[24] Daniel Fett, Ralf Küsters, and Guido Schmitz. "An Expressive Model for the Web Infrastructure: Definition and Application to the Browser ID SSO System". In: *2014 IEEE Symposium on Security and Privacy*. 2014, pp. 673–688. DOI: 10.1109/SP.2014.49.

[25] T. Lodderstedt, M. McGloin, and P. Hunt. "RFC6819 - OAuth 2.0 Threat Model and Security Considerations". In: *IETF* (2012). URL: https://tools.ietf.org/html/rfc6819.

[26] Eugene Ferry, John O'Raw, and Kevin Curran. "Security evaluation of the OAuth 2.0 framework". In: *Information and Computer Security* 23 (Mar. 2015), pp. 73–101. DOI: 10.1108/ICS-12-2013-0089.

[27] Ronghai Yang et al. "Model-based Security Testing: An Empirical Study on OAuth 2.0 Implementations". In: May 2016, pp. 651–662. DOI: 10.1145/2897845.2897874.

[28] T. Lodderstedt et al. "DRAFT - OAuth 2.0 Security Best Current Practice". In: *IETF* (2021). URL: https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics/.

[29] Hui Wang et al. "Vulnerability Assessment of OAuth Implementations in Android Applications". In: Dec. 2015, pp. 61–70. DOI: 10.1145/2818000.2818024.

[30] Ethan Shernan et al. “More Guidelines Than Rules: CSRF Vulnerabilities from Noncompliant OAuth 2.0 Implementations”. In: July 2015, pp. 239–260. ISBN: 978-3-319-20549-6. DOI: 10.1007/978-3-319-20550-2_13.

[31] Natsuhiko Sakimura et al. “Openid connect core 1.0”. In: *The OpenID Foundation* (2014), S3.

[32] Sven Hammann, Ralf Sasse, and David Basin. “Privacy-preserving openid connect”. In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 2020, pp. 277–289.

[33] Jorge Navas and Marta Beltrán. “Understanding and mitigating OpenID connect threats”. In: *Computers & Security* 84 (2019), pp. 1–16.

[34] Daniel Fett, Ralf Küsters, and Guido Schmitz. “The web sso standard openid connect: In-depth formal security analysis and security guidelines”. In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE. 2017, pp. 189–202.

[35] Wanpeng Li and Chris Mitchell. “Analysing the Security of Google’s Implementation of OpenID Connect”. In: July 2016, pp. 357–376. ISBN: 978-3-319-40666-4. DOI: 10.1007/978-3-319-40667-1_18.

[36] Christian Mainka et al. “SoK: single sign-on security—an evaluation of openID connect”. In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2017, pp. 251–266.

[37] Zhiyi Zhang et al. “EL PASSO: Efficient and lightweight privacy-preserving single sign on”. In: *Proceedings on Privacy Enhancing Technologies* 2021.2 (2021), pp. 70–87.

Block-chain based Payment Systems on the Cloud - Group 20

Abhishek Iyer Heymi Dannon Romano Smruti Inamdar

June 3, 2022

Abstract

With this paper, we try to give an insight on the current state of blockchain, cloud computing technologies and try to explain the importance of blockchain based payment systems that are powered by smart smart contracts and how the current research done into the same tells us that, these payment systems are better than the traditional payment systems that are currently in place, by giving an insight on how the problems that arise with use of traditional payment systems can be mitigated by blockchain based payment systems

1 Introduction

With this literature review, we aim to provide an in-depth insight on the research done in the area of cloud payments done using block-chain and the usage of smart contracts. With the rapid evolution of cloud computing, there has been a drastic increase in outsourcing services done over the cloud too, so it is of paramount importance to both the users and the cloud service providers to have a payment system in place that is reliable, scalable and most importantly, trustworthy. The main aim of blockchain is to decentralize, and this, in the payment systems over cloud is quintessential, as the current traditional payment systems in place on the cloud are riddled with issues like . So, in the literature review, we will be talking about various types of the systems implemented in various papers and we will do a comparative analysis of the various implementations and how well they work in order to eliminate the need for a third party system like banks.

We target to answer the following question(s) "What are the issues with the traditional payment systems on the cloud? How can the use of block chain and smart contract improve this? What is the scope of such a system?"

This paper is structured in the following way. We first give a brief introduction to blockchain technology and then dive deeper into its architecture, applications and the challenges that arise with it. Then in the subsequent section we talk about cloud computing and how it can facilitate blockchain as a service and the

challenges faced by cloud computing. Then, we talk about Blockchain and cloud and how the two technologies are being used together to mitigate the issues faced by each of them and we also talk about smart contract in that section and what they really are. We then proceed to talk about various blockchain based payment systems that have been implemented in the cloud and discuss how they are being used to mitigate the issues that arise with using traditional payment systems. Finally, we talk about the future work and scope of these systems that me mention in preceding sections and arrive at a conclusion about our research question,

2 Blockchain fundamentals

Distributed ledger technology is a decentralized database managed by multiple users [19]. Blockchain is a digital ledger of transactions that are distributed across an entire network of computers. It is a method of recording information which becomes immutable. Blockchain is a special form of DLT where transactions are recorded using an immutable hash.

Bitcoin, a very popular decentralized peer-to-peer digital currency, makes extensive use of the Blockchain technology [4]. The fundamental idea of blockchain revolves around creating a digital consensus and this is achieved with giving utmost importance to anonymity. The inception of bitcoin was done with the paper "Bitcoin: A Peer-To-Peer Electronic Cash System". It describes electronic cash transactions done without the presence of a trusted third party or financial institution. Every transaction is protected using a digital signature. Following summarizes important steps followed by a transaction:

- Private key of the sender is used to digitally sign the transaction sent to the public key of the receiver.
- The digital signature is then verified by the entity receiving the digital currency.
- Transactions are broadcast to every node of the bitcoin network, are verified and then recorded in a public ledger.

The verifying node has the responsibility to check if Sender owns the cryptocurrency and if sufficient crypto balance is present in the Sender's account. Their Public key needs to be verified in the ledger.

2.1 Consensus Algorithms

To add a block into the Blockchain, the block has to be verified by all nodes on the network. Consensus algorithms are a bunch of protocols that maintain order of the blocks to be added and check for validity of existing blocks. In a real time scenario, more than one blocks are competing to be added into the chain. Consensus algorithms step in and help with decision making as miners

cannot reach a consensus without a central authority.

Transactions are not always in the order of generation. Since bitcoin is a distributed system, it is a daunting task to maintain consensus on the order. This problem is solved using Blockchain Technology. Blocks are groups of transactions which are linked together, in a linear and chronological order. Every block contains a hash of the previous block, it's encrypted hash and a timestamp.

2.1.1 Proof of Work

Multiple blocks could be generated at different nodes at the same time. There is a possibility of unverified transactions to be broadcasted to the remaining network. To identify the order of blocks' placement, a user needs to solve a computationally challenging problem. The network node uses a block header and a nonce (4-byte field beginning at zero) to obtain a cryptographic hash function SHA-256. When the target value is achieved, the miner block is broadcasted to all the network nodes. Validity of the hash value has to be then confirmed for approval of the new block. It is then connected to the blockchain. This system is called Proof-of-Work.

Another tricky scenario stems when miner blocks are generated in parallel. The Proof-of-Work protocol follows the rule of longest chain, where the longer the chain is, the more authentic it is. Miners end up using a lot of computation power in PoW, which results in resource wastage. [4]

2.1.2 Proof of Stake

Proof-of-Stake is an excellent alternative to PoW, and deals with resource wastage. Miners with more coins are more likely to have their block inserted in the Blockchain system. This leads to a bias towards the wealthiest miner. Solutions like Peer coin are used, where the age of the coin is favored for selection of the next block. Blackcoin is yet another solution which uses randomization to predict the next block.

2.2 Blockchain Architecture

Blocks : Blockchain comprises of fine-grained components called blocks which are linked together in a linear chain fashion. Every block comprises of transaction ID provided by the users, Block height, Hash values of previous and the current block, Merkle Tree Root hash, Timestamp, block size and list of transactions.[6]

Mining Nodes : They produce Blocks to add to the Blockchain. Mining nodes are not responsible for maintenance of blocks. Blocks produced are transmitted throughout the network and then verified by participant nodes for authenticity.[3] Images 1 and 2 show the architecture of a blockchain and dissection of a block, respectively

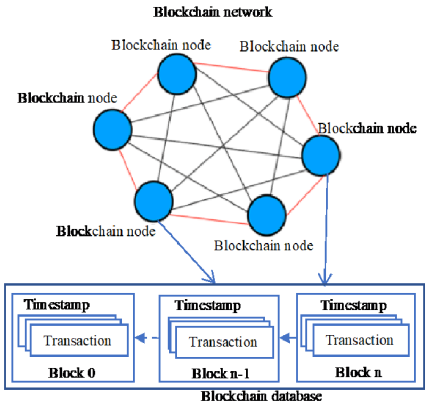


Figure 1: Blockchain Architecture

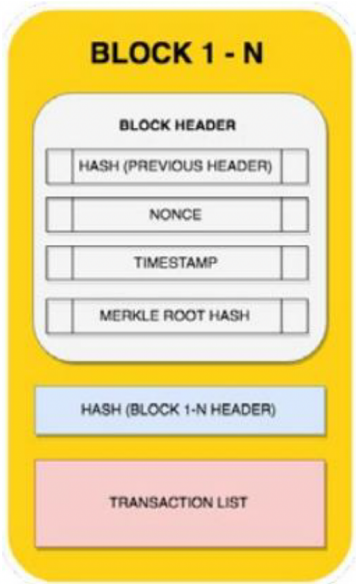


Figure 2: Dissection of a Block

2.3 Applications

Fintech times made an elaborate study on the various use cases of Blockchain, ranging from Crypto to Central Bank Digital Currency (CBDC). The idea was to emphasize on it's applications other than crypto. Blockchain has been used to make payments to buy digital artwork (NFT) , or even as simple as a morning coffee. Of late, banks have been issuing cards which deal with transactions with cryptocurrency balance. This has broken barriers of traditional payment methods, while encouraging cross-border payments. [10]

2.4 Challenges in Blockchain

Like any other technology, blockchain has it's challenges and shortcomings.

Privacy: Transactions are stored in a public ledger and are made with anonymous addresses in place of real identities. Blockchain doesn't assure privacy, as the value of transactions is available to the public key.Privacy is an essential requirement for cryptocurrency. Using a private or federated blockchain is a good workaround for privacy. [3]

Scalability: A blockchain is made of blocks and the size of blocks are fixed. With an increase in the users, the block generation time also increases. This leads to higher and time consuming transactions, which defeats the purpose of using Blockchain.

Higher energy consumption: Blockchain infrastructures like bitcoin use

constructs like Proof-of-Work and consensus algorithms. The mining process requires a lot of computational power. It is important to maintain a balance between blockchain architecture and applications. But we can't allow Bitcoin and PoW mining's potential heavy energy consumption to sidetrack blockchain's potential upsides — again; high energy consumption is only inherent to Bitcoin (and other PoW Cryptocurrencies to a minor degree).

3 Cloud Computing

Cloud Computing is the delivery of services like storage, databases, software and networks over the internet. It has emerged from distributed computing technology. A major advantage of this technology is that the user can only pay for the services used, hence leading to lower operational costs and efficient infrastructure. [6]

3.1 Delivery models of Cloud Services

- **SaaS**: **Software as a service** is where the whole application is deployed on the internet. These services are also accessed using a dashboard like Iaas. Saas leverages on maintenance time with control and security. Examples: Gmail, Salesforce, Amazon Web Services
- **PaaS**: **Platform as a service** is where the cloud service provider allows a user to deploy applications within the platform. Examples: Google Search Engine
- **IaaS**: **Infrastructure as a service** is a pay-as-you-go service where the user has direct access to processing ,storage and software across the internet .These services are accessed and controlled using an API. There is flexibility to purchase only the services needed. Virtualization is used to distribute resources as per cloud user demands. Examples: Amazon EC2,GoGrid [9]

3.2 Blockchain as a Service

An extension of the above is **Blockchain-as-a-Service (BaaS)**. It is a third party cloud-based infrastructure for building blockchain apps and is based on SaaS. [8] Organizations can access the blockchain service built on the cloud. And like traditional Blockchain applications, this also consists of smart contracts. Cloud based service provider is responsible for the maintenance of the infrastructure. [8]

In the Blockchain as a Service (BaaS) model, businesses and organizations can access the Blockchain service that is created and developed on a cloud. A blockchain as a service application is developed, hosted and deployed on the cloud. This application is like any other natively hosted blockchain application

that has smart contracts and other relevant Blockchain functions. The advantage of a blockchain as a service (BaaS) model application is that the business need not worry about the management and installation of any kind of infrastructure like the server and instead depend on the cloud-based service provider to do all these IT-related jobs. Examples: Azure, AWS, R3 Corda

3.3 Downsides of Cloud

Downtime: It is one of the biggest downsides of Cloud computing. It being based on the internet, there are possibilities of network and service outages due to bottlenecks, power outages, technical difficulties, slow internet connections,etc.

Data Encryption: Encryption of data is done to provide an extra layer of security. Cloud largely uses SSL encryption for access. It is well known that data is decrypted before storing in the cloud. This leads to data security being compromised while storing decrypted data without prior encryption.

Vendor Lock-in: Migration of data and applications from one cloud vendor to another isn't an easy task due to it's rigid infrastructure. Even if the migration is successful, it could lead to support issues and configuration complexity. The user data might be left vulnerable and prone to attacks due to compromises made during migration.

Limited control: Users don't have a complete control on their deployed applications. Cloud Services on remote servers are owned, deployed and managed by Cloud service providers, leaving little room for the users to make any modifications to the backend infrastructure.

4 Blockchain and Cloud

4.1 Integration of Blockchain and Cloud

Sections 2 and 3 give a brief overview of Blockchain and Cloud systems. This section focuses on the integration of Blockchain with Cloud Computing. Blockchain provides solutions like data privacy, network security and interoperability to mitigate the downsides of Cloud mentioned in Section 3. Cloud computing is made of large networks of virtualized resourcces, hardware and software resources. Blockchain can be integrated with cloud systems to facilitate data replication, easier access to transactional databases and storage. Security and privacy are important features offered by blockchain which makes user, task and data management easier. [16]

4.2 Smart Contracts

"A smart contract is the formalisation of an agreement, whose terms are automatically enforced by relying on a transaction protocol, while minimising the need of intermediaries."(Scoca,Uriarte,De Nicola, 2). "Smart contracts are computer

programs that can be consistently executed by a network of mutually distrusting nodes, without the arbitration of a trusted authority. Being embedded in blockchains, smart contracts enable the contractual terms of an agreement to be enforced automatically without the intervention of a trusted third party."(Jani, 2)

Simply put, smart contracts are just programs stored in a blockchain which are used to automate the execution of agreements between the user and the service providers, when the predetermined conditions are met at both the user and server side. They play an important role in the blockchain payment systems as they replace the need for an intermediary, it the smart contract's job to constantly keep checks on both the user side and the server side and make sure that the pre-determined conditions are being met from both ends. In case the smart contract notices a lapse on either side, based on the implementation of it within a framework, it will either make suggestions to the side where the conditions are not being met by either sending error messages or simply refusing to execute an agreement. It is also used to hold in information like public keys and digital signatures in order to maintain the information of the user and the server side.

4.2.1 Interoperability

Internal communication is not allowed with Public clouds, but with an addition of blockchain technology to the cloud, seamless inter-node communication is possible. Each cloud entity is treated as a node and every node on the network contains a copy of transactions.

4.2.2 Cloud Data Management

The data stored in cloud is unstructured as opposed to structured data on Blockchain. Since every block contains a hash from the previous block, it is easy to locate required data. Since Blockchain is a Distributed ledger, large number of transactions can be processed and stored easily. This extends to involvement of smart contracts, which increased service quality. [12]

4.2.3 Data Encryption

It is known that data is decrypted before storing in the cloud. Blockchain network has block data which uses cryptographic algorithms to transform into hash keys. Block data integrity is of utmost importance and is achieved using block discovery consensus mechanisms. Every node on the network has a copy of the transactions, which means easy availability. This also improves persistence which helps network deal with security attacks.

4.3 Tassat Case Study

Tassat Group, a New York based company provides Blockchain technologies and solutions for financial institutions. They deal exclusively with **Business-to-Business (B2B)** payments. Recent digital economy has setbacks like processing

delays, transaction limits and exorbitant transfer fees. This resulted in the need for real-time payment processes. Most banks lack infrastructure and resources to build technology to process legacy payments. Tassat group focused on creating a Private Blockchain-based payment platform for it's customers. This technology was integrated with cloud making a labor-intensive and expensive approach redundant. the integrated model is what became TassatPay. [11]

Table 1: Summary of infrastructure functionality and technology used

Functionality	**Technology Used**
Credential Managament	Google Cloud Secret Manager
Scalability	Google Cloud Compute Engine
Storing and Managing Data	Google Cloud Filestore
Web access optimization	Google Cloud Load Balancing
Guided Design Work	Google Cloud Architecture Framework
Visualization of Energy Impact	Google Cloud Carbon Footprint

The TassatPay runs on a microservice based architecture build around a private, permissioned Blockchain and a message bus. Compute Engine was chosen to achieve speed and scalability. It aids with performance considerations like memory-optimized workloads and general purpose computing. 1 summarizes the functionality required for the Blockchain-Cloud infrastructure and the Technologies issued by Cloud service provider, which in this case, is Google.

4.3.1 Outcomes

Using Tassat infrastructure with Google cloud aided in reducing energy consumption associated with transaction. A majority of regulatory requirements of the finance industry like privacy, security and others are met using above technologies.

5 Analysis of various payment systems

5.1 BCPay

This is a payment framework that was introduced in [18], the main aim of this framework is to eliminate the need for a trusted third party to build trust between the users and the cloud service providers. This framework has robust fairness and soundness. i.e., it is quite resilient to attacks like eavesdropping and also that is is quite efficient.

It achieves it's robust fairness and soundness using a protocol called the all or nothing checking proof protocol [18]. This protocol aids the service providers to guarantee back immediately after the user has paid the fee for the required service, or, it is penalized in form of a deposit payment to the user.

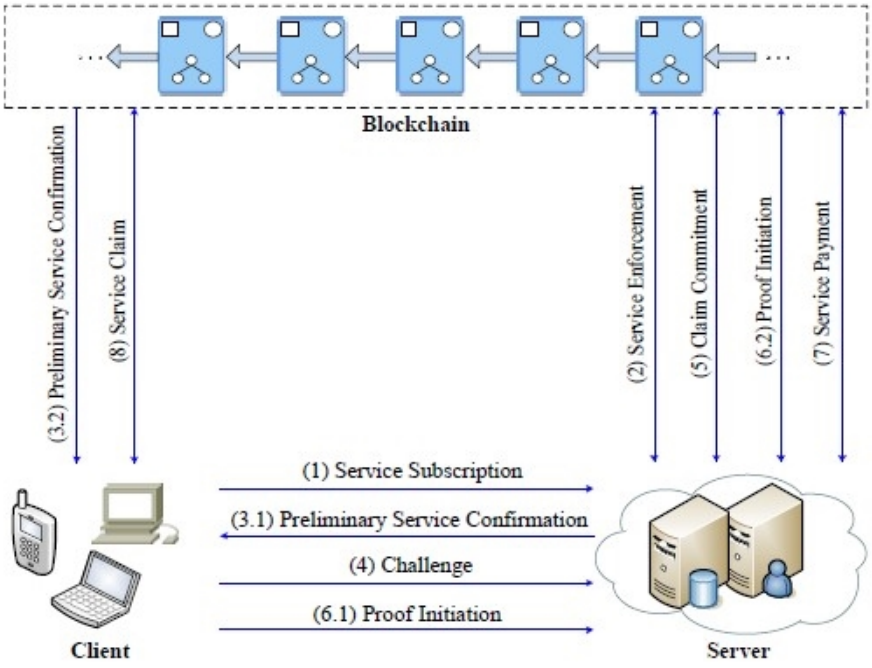


Figure 3: BCPay Architecture

Image 3 shows the architecture of BCPay as described in [ref]. There are three key actors in the BCPay architecture, i) Users i.e Client, ii) Cloud Service Providers i.e Server and iii) Blockchain. The user, wants to gain access or subscribe to a particular service that is offered by the server and the server wants to obtain the fee for the service being provided and also send the user back with the access to the said service as well as confirmation of the subscription fee. Meanwhile, the blockchain has which has a smart contract, checks if the conditions at both ends are met, when they are, the agreement is then executed. With this, BCPay succeeds in eliminating the drawbacks of the traditional payment system like the need for a trusted third party i.e., a bank and the assumption that it is indeed trusted by both the user and service provider, the bottleneck that is cause when any participant changes their bank and then that bank or third paty needs to registered and also the issue of user's privacy being violated by the third party. BCPay, whilst mitigating the drawbacks of the traditional system, also speeds up the payment process as the computational time lost due to having an intermediary, is no longer an issue.

BCPay's functionalities can be summed up in five phases. The first one being the **System Setup Phase**. This is the phase where both parties (the user and

the service provider) set some important parameters on the blockchain, this is done inorder to define the conditions the need to met inorder for the transaction to be considered proper and complete by the smart contract that is stored in the blockchain. The second phase is the **Service Implementation phase**, in this phase the service that is requested by the user is implemented. In this phase, the user subscribes to the service provided by the service provider and sends the data relating to the service is sent by the user to the service provider, on receiving that data, a digital signature is created and stored in the blockchain by the service provider and then the service provider sends a confirmation message that helps the user get the signature from the blockchain and on obtaining the signature, the user is assured that the service they have subscribed for is implemented (which is checked by the user before the subsription fee is paid). The third phase is the **Service Checking Phase** this phase is jointly initiated by both the user and the service provider. In this phase, the conditions of the service that is implemented are checked and both the user and the server come to agreement and if both parties are honest, then the user gets the assurance the service will be as per the agreement and the service provider will get the subscirption fee from the user. The fourth phase is the **Service Payment Phase** in this phase, the user pays the subcription fee to the service provider after checking that the service implementation meets the predetermined criteria. The final phase is the **Service Claim Phase** BCPays only enters this phase if the service provider fails to prove that the service requirements are met, before a specific time. In this phase, the user can claim enough deposits from the service provider.

5.2 BlockSubPay

This is payment system mentioned in [5], it is influenced by BCPay. Unlike BCPay, in this framework, two ways of subscription are taken into account as usually some cloud service providers work with a fixed subscription fee model wherein, a fixed fee is levied for the service that a user wants to use. Then there is a pay as you go model, where the payment depends on factors like how much computation is done and how much memory is used. In that case the user is billed according to their usage.

In this framework the authors propose a protocol that has four stages. (i) Subscription Registration Stage, (ii) Access Token Request Stage, (iii) Cloud Resource Access Stage, and (iv) Subscription Expiration Stage. In the first stage, the user selects the service that they want to subscribe to, from a particular cloud service provider and on selecting the particular service and the service provider provides the user with necessary information regarding that particular service, which includes the subscription fee and the address to the smart contract which is to be accessed by the user to make the fee payment. When the payment has been made by the user, the smart contract checks if it meets the all the criteria for a successful payment, after doing the checks, if the smart contract finds no fault in the payment, it records the user's public key address which was used to transfer the payment into the pool list of public keys and then it

also sends out valuable information to the user like the subscription status and expiry date of the subscription. In pay as you go model, the subscription fee is usually a minimum deposit made by the user, which guarantees that the user is honest and genuine and in that model, the smart contract also returns valuable information like how much of the deposit money has been used the user and how much more computations the user can do before all the money has been used. In the second stage, after the payment of the subscription fee, the user asks the service provider for an access token in order to access the servers. The service provider, on the request of the user, will open a gateway, through which the user signs his requests with the public key used during the payment of the subscription fee (or in the case of a pay as you go model, the deposit). On receiving the request, the cloud service provider uses the smart contract to make checks on the public key of the user and if the public key is found in the pool of other public keys and also if the subscription status is valid, then access token is granted to the user. In the third stage, the user wishes to access the server of the cloud service provider using the access token that was granted to them in the previous stage. The user accesses the gateway and sends in a request with using the access token they have, on receiving the request, the smart contract checks the access tokens to see if they are valid and if they are meant to access the server that the user wishes to access, along with those checks, it also checks the subscription status of the user, if all is in order, then the access is granted to the user. The final stage occurs when the user's subscription status turns to inactive. In this case, the user can either choose to extend the subscription by paying the subscription fee again, or in the case of pay as you go model, by making a deposit again and resuming their subscription. The user can also upgrade or downgrade by choosing the service which best suits their current need. The user can also choose to not do anything and let the subscription be inactive and discontinue using the services provided by that cloud platform.

6 Future Scope & Work

Today, various industries are reliant on mobile payment or digital transactions that require verification. From using the cloud service provider for verification stage of the payment protocols to using smart contracts that settle the service agreements for the cloud service requirements or the energy market [14] to even auditing the integrity of a cloud service, blockchain technology coupled with cloud architecture and containerization poses great solutions for the modern issues that arise with new systems.

One of the main issues in mobile payment architecture [2] is the trustworthiness of the cloud service provider that provides the computation that is required to validate the pseudo keys used in the transactions. CSPs are prone to colluding with each other for profits. However, on the contrary to the efforts of the paper, smart contracts can be used to audit the CSP's integrity. The articles [20] and [13] shows that with minimal overhead, the third party auditor

can be replaced. As the paper indicates, the overhead of ProofGen and Verify is approximately 14 ms and 11 ms respectively, when the number challenge blocks reaches 1000. These results show that we can have the CSP auditing automated using blockchain technology in a scalable manner which fixes the trustworthiness issue around the CSP involved in the payment process.

A great issue with acquiring cloud services is to agree on a Service Level Agreement. This contracts essentially depicts what the CSP agrees to provide and what the requestor demands. However, this contract needs to be dynamic as the demands of the requestor adjust according to various factors. For example, Netflix would require different resources for streaming video to its customers for different segments of video quality and the video quality is subject to change as the user upgrades or downgrades their subscription plan. Smart contracts can provide this exact service as they are software that can automate the construction and negotiation of the terms between the requestor and the service provider. Using blockchain technology, the validity of the contracts can be proven and the older contracts can be acknowledged in the process as well. The article [7] demonstrates a clear design of a system that automates the process of finding the suitable SLA [1] for the requestor in a fast, automated way. As flexibility is adjustable, the results show that for certain level of thresholds, autonomous negotiation system can achieve more matches than a classic bilateral negotiation.

Energy market is another sector where cloud services and the smart contracts can completely digitize the who paradigm. In the energy market, energy providers buy or sell as the demand and supply for energy(ie. carbon) shifts constantly. These transactions are crucial to maintain the equilibrium in the demand and supply for energy. The article [14] outlines a design of a cloud based platform that leverages the use of smart contracts to meet the dynamic nature of the energy transactions. The design was tested for different scenarios in which natural gas contracts were bought and sold using the Antchain contracts rather than Etherium because Antchain costs only 0.25 yuan to compute while Etherium contracts computational cost can go up to 12.71 yuan. The article depicts a viable system which was developed using Cloud IDE provided by Blockchain-as-a-Service (BaaS) platform.

On the more general use cases of smart contracts, Stellar emerges as a cost effective alternative. Stellar is able to support languages such as Python, JavaScript, GoLang, or PHP. Stellar achieves the cost of effective execution by using Docker containers. To be precise, the execution cost of one transaction at Stellar is only $ 0.0000002 and the execution time is around 5 seconds. With such low cost and high speed, Stellar technology can digitize the process of creating, negotiating, and validating contractual requirements of different payments in scalable manner.

7 Conclusion

From the research that we conducted in the area of blockchain, cloud computing, smart contracts and various blockchain based payment systems that use smart contracts, that are being implemented on cloud, we can come to the conclusion that these payment system frameworks certainly do help in mitigating various issues like user's data privacy, deduplication [15], high computational time and cost. To add to this, blockchain and smart contract powered payment systems can also be extended to various types of subscription model like fixed subscription as well as pay as you go model. They are also quite safe and reliable and don't suffer from issues like eavesdropping.

Albeit new, these systems can replace the traditional payment systems and not just in a single user and server case,these systems can be extended to sectors like the energy sector, where we deal with multiple users and multiple service providers [14][17]. Since, the smart contracts in blockchain can keep a log of public keys and store it, it can be extended to facilitate a scenario where multiple users are at the user level and multiple service providers are the server level for a single scenario. Such cases, where traditional payment systems could cause a bottleneck when multiple participants on any either end change their trusted third parties and work to establish trust. It is also useful as having the blockchain layer with smart contracts can be used to store pre-determined conditions for all the participants on both ends.

Hence, it is safe to say that continued research and development in this field of study can help a lot of sectors to transition from traditional payment mechanisms and adopt a blockchain based payment system which are powered by smart contracts.

References

[1] Wenjuan Li et al. "Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions". In: *Journal of Cloud Computing* 10 (June 2021). DOI: 10.1186/s13677-021-00247-5.

[2] Yongjian Liao et al. "Analysis of a mobile payment protocol with outsourced verification in cloud server and the improvement". In: *Computer Standards Interfaces* 56 (Sept. 2017). DOI: 10.1016/j.csi.2017.09.008.

[3] Ch Murthy et al. "Blockchain Based Cloud Computing: Architecture and Research Challenges". In: *IEEE Access* 8 (Nov. 2020). DOI: 10.1109/ACCESS.2020.3036812.

[4] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Cryptography Mailing list at https://metzdowd.com* (Mar. 2009).

[5] Yustus Oktian et al. "BlockSubPay - A Blockchain Framework for Subscription-Based Payment in Cloud Service". In: Feb. 2019, pp. 153–158. DOI: 10.23919/ICACT.2019.8702008.

[6] s Sarmah. "Application of Blockchain in Cloud Computing". In: *International Journal of Innovative Technology and Exploring Engineering* 8 (Oct. 2019), pp. 2278–3075. DOI: 10.35940/ijitee.L3585.1081219.

[7] Vincenzo Scoca, Rafael Brundo Uriarte, and Rocco De Nicola. "Smart Contract Negotiation in Cloud Computing". In: May 2017. DOI: 10.1109/CLOUD.2017.81.

[8] Blockchain Simplified. *Blockchain as a Service*. 2021. URL: https://medium.com/@blockchain_simplified/the-implementation-of-blockchain-as-a-service-baas-de18490c9887.

[9] Blockchain Simplified. *Delivery Models*. 2020. URL: https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas?sc_cid=7013a000002pgRPAAY&gclid=Cj0KCQjwnNyUBhCZARIsAI9AYlEuk9iKFt1m%20JtT6hS2mqSbcyvoe3LgOjDj-3F8nG5VHfCFC_3TIy9kaArWVEALw_wcB&gclsrc=aw.ds.

[10] Tyler Smith. *Financial Times Study*. 2020. URL: https://thefintechtimes.com/blockchain-how-its-being-used-to-process-payments/.

[11] *TassatPay*. 2020. URL: https://cloud.google.com/customers/tassat.

[12] Karthikeya Thanapal et al. "Online Payment Using Blockchain". In: *ITM Web of Conferences* 32 (Jan. 2020), p. 03007. DOI: 10.1051/itmconf/20203203007.

[13] Hao Wang et al. "Blockchain-based fair payment smart contract for public cloud storage auditing". In: *Information Sciences* 519 (May 2020). DOI: 10.1016/j.ins.2020.01.051.

[14] Lei Wang et al. "Design of integrated energy market cloud service platform based on blockchain smart contract". In: *International Journal of Electrical Power Energy Systems* 135 (Feb. 2022), p. 107515. DOI: 10.1016/j.ijepes.2021.107515.

[15] Shangping Wang, Yuying Wang, and Yaling Zhang. "Blockchain-Based Fair Payment Protocol for Deduplication Cloud Storage System". In: *IEEE Access* PP (Sept. 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2939492.

[16] Zhiran Wang. "Research on Cloud Computing-Based Online Payment Mode". In: *2011 Third International Conference on Multimedia Information Networking and Security* (2011), pp. 559–563.

[17] Wei Wu et al. "Blockchain-based smart payment scheme of power infrastructure funds". In: *Energy Reports* 7 (Nov. 2021), pp. 725–733. DOI: 10.1016/j.egyr.2021.09.199.

[18] Yinghui Zhang et al. "Blockchain based Efficient and Robust Fair Payment for Outsourcing Services in Cloud Computing". In: *Information Sciences* 462 (Sept. 2018), pp. 262–277. DOI: 10.1016/j.ins.2018.06.018.

[19] Zibin Zheng et al. "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends". In: June 2017. DOI: 10.1109/BigDataCongress.2017.85.

[20] Jinglin Zou et al. "Integrated Blockchain and Cloud Computing Systems: A Systematic Survey, Solutions, and Challenges". In: *ACM Computing Surveys* (Mar. 2021). DOI: 10.1145/3456628.

Distributed neural graphs: DistDGL vs GraphTheta vs AliGraph

Effrosyni Stergiopoulou
Department of Computer Science
University of Amsterdam
13915681
effrosyni.stergiopoulou@student.uva.nl

Noé Zhang
Department of Computer Science
University of Amsterdam
14149117
noe.zhang@student.uva.nl

Anastasios Bazinis
Department of Computer Science
University of Amsterdam
14206587
anastasis.bazinis@student.uva.nl

Abstract

This study focuses on graph neural networks (GNNs) and analyzes 3 state-of-the-art methods that can be directly applied on graphs to perform inference. In the first section, we introduced GNNs by formalizing them, describing basic methods required such as partitioning, sampling, and training procedure, and addressing challenging aspects that this field currently faces based on recent bibliography. In the following chapters we describe three of the most prominent GNN learning methods, namely AliGraph, DistDGL, GraphTheta. We provide an overview of their basic attributes, system architecture and methods used or supported. Following that analysis we provide a comparative analysis of the three methods in terms of their most important features in each of the basic components(partitioning,sampling,training). Finally, we provide a qualitative evaluation of the three methods concerning their scalability potential. According to our analyis, we came to the conclusion that GraphTheta appears to be the dominant method among the three discussed since it is tested under a real world size dataset and appeared to outperform existing methods and scale linearly in comparison to the workers/nodes provided in a distributed setting(Linux CPU Docker). Finally, our conclusion offers some future work suggestions as well as methods that could be enhanced to produce even more efficient results.

1 Introduction

Over the last several years, graph neural networks (GNNs) have demonstrated exceptional performance in a range of graph-based applications, including social networks and recommendation systems. They are a rapidly growing graph learning family of deep neural network designs for which, common DNN practices like data and model parallelism, do not apply (1). Despite recent advances, training GNNs on large-scale real-world graphs remains challenging. There has been several attempts to expand GNN training utilizing sampling methodologies however, they are still inefficient for training on extremely large graphs due to the unique structure of GNNs and the limited memory capacity of present servers. One alternative is to use distributed training with data parallelism as a potential solution to these limitations as moving from a single machine to several machines reduces the training time and memory burden on each unit.

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

However, due to the reliance between nodes in a graph, generalizing the existing data parallelism approaches of conventional distributed training to the graph domain is not straightforward. GNNs, for example, rely significantly on the information intrinsic to a node and its nearby nodes, unlike picture classification issues where images are mutually independent, allowing us to divide the image dataset into various partitions without worrying about image dependence. As a result, partitioning the graph produces subgraphs with edges crossing subgraphs, causing information loss and slowing down the model's performance. To address this issue, multiple approaches have been proposed such as transferring node features between local machines however, it can result in considerable storage or communication overhead and privacy problems.

For graph analytics, machine learning and deep learning are becoming more popular. Graph Neural Networks (GNNs) are a new type of Deep Neural Network (DNN) architecture designed specifically for graphs, in which the neural network's structure overlaps with the graph's structure(2; 3). A GNN is made up of layers that transform the inputted features into a set of outputted features in the form of embeddings. Embedding is a process of converting high-dimensional data into a lower-dimensional latent representation like vectors. These representations make it easier to do machine learning such as prediction and classification tasks on large or sparse inputs like the ones from graph neural networks.

1.1 GNN learning approaches

The primary idea behind node embedding methods is to employ dimensionality reduction techniques to convert high-dimensional information about a node's network surroundings into a dense vector embedding. These node embeddings may then be supplied into downstream machine learning systems to help with node classification, clustering, and link prediction, among other tasks(4).

There are two major types of learning approaches, the transductive and the inductive. When we use transductive learning, we can infer only the labels of unlabeled nodes that were in the training set, while for inductive learning we train a predictive model M that can dynamically predict labels of nodes that were not in the training set.

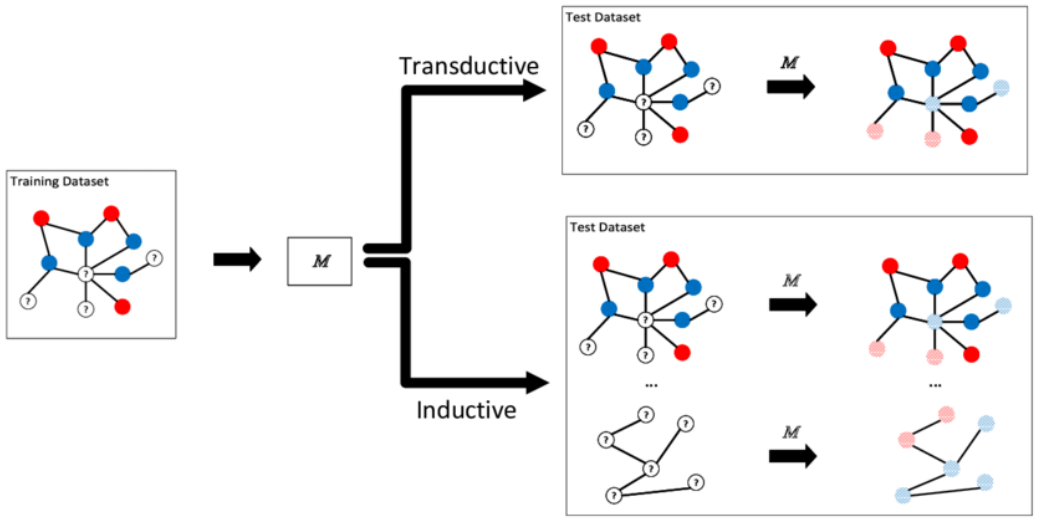


Figure 1: Comparison between transductive and inductive setting(5)

1.1.1 Transductive Graph Learning

The majority of current methods for producing node embeddings are intrinsically transductive. Because they make predictions on nodes in a single, fixed graph, most of these algorithms explicitly optimize the embeddings for each node using matrix-factorization-based objectives and do not typically extend to unseen data(6).Transductive GNNs create a model based on the data points collected during training and testing but it does not build a predictive model that could be generalizable. Using the knowledge of the labeled points and extra information, this method predicts the labels of unlabeled points but if new, unlabeled data points are met typically we will have to rerun the algorithm

from scratch. These algorithms may be tweaked to work inductively to infer new knowledge from the graph, but doing so is computationally costly, necessitating more rounds of gradient descent before new predictions can be produced(7).

1.1.2 Inductive Graph Learning

Previous research has concentrated on embedding nodes from a single fixed graph, however many real-world applications demand embeddings for unseen nodes or totally new (sub)graphs to be produced fast. For high-throughput, production machine learning systems that work on developing graphs and frequently meet unseen nodes, this inductive power is critical (e.g., posts on Reddit, users and videos on Youtube). An inductive approach to generating node embeddings also makes it easier to generalize across graphs with similar features: for example, one could train an embedding generator on protein-protein interaction graphs derived from a model organism, and then use the trained model to easily generate node embeddings for data collected on new organisms(8).

During the training phase of inductive learning, the nodes that will be utilized for testing are completely hidden. When compared to the transductive approach, the inductive node embedding case is extremely complex since generalizing to unseen nodes entails "aligning" unseen subgraphs to the node embeddings that the algorithm has previously optimized. An inductive framework must learn to detect structural aspects of a node's neighborhood that disclose both the node's local and global significance in the graph. An inductive framework must learn to detect structural aspects of a node's neighborhood that disclose both the node's local and global roles in the graph.

1.2 Partitioning of Graphs

How information spreads throughout the graph is a critical topic. There are several cases in which information must be transmitted across long distances across a network, such as when we have long sequences reinforced with extra relationships between the sequence's constituents, such as in the text, computer language source code, or temporal streams. The most basic strategy, and one used by nearly all graph-based neural networks, is to emulate synchronous message-passing systems from distributed computing theory (9). In particular, the inference is carried out in a series of rounds in which each node sends messages to all of its neighbors, the messages are delivered, and each node does some calculation depending on the received messages. While this method is straightforward and fast to apply, it is inefficient when the task demands spreading information across large distances in the graph.

As a result, extensive graph partitioning methods are utilized in GNNs to attain a better degree of efficiency and parallelization of calculations. Graph partitioning is a method of dividing a graph into smaller sections in which the nodes are mutually exclusive. It is an efficient method for reducing complexity or parallelizing, and the partitioned graph becomes more suitable for analysis and problem-solving. With the introduction of ever-larger instances in applications such as scientific simulation, social networks, and road networks, graph splitting becomes increasingly crucial, multidimensional, and difficult. Because graph partitioning is a difficult topic, several strategies and solutions are presented. Several heuristics and high-quality graph partitioning algorithms are among these solutions. The multilevel graph partitioning technique is the most successful heuristic for dividing huge graphs. It is divided into three stages: coarsening, initial partitioning, and uncoarsening(10).

1.3 Sampling techniques: Global-batch, Mini-batch, Cluster-batch

Global-batch and mini-batch are two prominent training algorithms on huge graphs from the standpoint of inductive learning. By combining feature matrices with graph Laplacian, global-batch executes globalized graph convolutions throughout an entire graph. Regardless of graph density, it requires calculations on the full graph(11). Mini-batch, on the other hand, performs localized convolutions on a batch of subgraphs, where a subgraph is made up of nodes and their neighbors(8).Subgraph building is frequently confronted with the problem of size explosion, especially when the node degrees and exploration depth are both big. As a result, training on dense networks (high-density graphs) or sparse graphs with strongly skewed node degree distributions is challenging. A number of neighbor sampling strategies are offered to lower the size of neighbors to lessen the issue of mini-batch training(12). However, the sampling strategies are frequently the reason behind unreliable inferencing results.

Cluster-batch, like mini-batch, trains a model on the subgraphs created by the first batches of target nodes, whereas global-batch trains a model on the whole graph. The number of GNN layers, graph topology, and community recognition techniques all influence the size of a subgraph, which can range from one node to the whole graph. The number of GNN layers controls the neighbor exploration depth, and subgraph sizes rise exponentially. Node degree affects the exponential factor of subgraph development in a graph topology. By utilizing community discovery (graph clustering) methods, this strategy may optimize per-batch neighborhood sharing and in some applications, it outperforms mini-batch.(13)

1.4 Formalization of Graph Neural Networks

According to Hu et al.(14), a GNN may be perceived as a message passing system that uses the input graph structure as the computation graph, with local neighborhood information collected to provide a more contextual representation(15).

Suppose H_t^l is the representation of node t at the (l)-th GNN layer, the recurring update procedure from the layer $(l-1)$ to the (l)-th layer is:

$$H_t^{(l)} \leftarrow \underset{\forall s \exists N_t, \forall \varepsilon \exists E_{(s,t)}}{Aggregate} \ (Extract(H_s{}^{(l-1)}; H_t{}^{(l-1)}, e)) \tag{1}$$

where $N(t)$ represents the source nodes of node t and $E_{(s,t)}$ represents the edges from node s to t. The most used operators for GNNs are **Extract** and **Aggregate**. The **Extract** operator stands for extracting neighbor information, using the target node's representation H_t^{l-1} and the edge e that connects the nodes as query, extracting information for the new node H_s^{l-1}. The **Aggregate** operator performs neighborhood information aggregation. Each aggregator function collects data from a certain number of hops away from a particular node, or search depth. We employ our trained system to construct embeddings for completely unseen nodes at test or inference time by applying the learned aggregation methods. The simplest aggregation operators are commonly regarded the mean, sum, and max functions, however advanced pooling and normalizing functions can also be developed. In principle, **Extract** operator is used by attention-based models assessing the significance of each source node, which is then returned to do a weighted aggregation.

1.5 Training with sampling algorithms

In practice, learning large-scale graph data necessitates significant computing and memory resources, resulting in a high training cost for GNN. To address the issue, sampling techniques for efficient GNN training are developed. Sampling methods decrease the computational and memory costs of training while maintaining acceptable accuracy loss by conditionally picking nodes(16).

Training GNNs, particularly GCNs, typically necessitates the implementation of the entire graph Laplacian and all intermediate embeddings, which incurs a high memory cost and makes it difficult to scale training on massive graph data. Furthermore, the standard training technique updates the model using a full-batch strategy, which results in a sluggish convergence rate.To tackle these shortcomings, sampling algorithms are proposed that alter traditional training by using a mini-batch approach and conditionally selecting partial neighbors, decreasing memory and computation costs. According to the granularity of the sample operation in one sampling batch, sampling algorithms may be classified into a variety of categories(17).

The respective categories are node-wise, layer-wise, and subgraph-based.The typical node-wise sampling algorithms,treat several hops, randomly sampling each node's neighbors. Layer-wise sampling algorithms use top down sampling on a multi-layer model. They focus on a fix number of sampling nodes per layer without taking into consideration the number of the nodes' neighbors. In subgraph-based sampling techniques, one subgraph is sampled for each mini-batch for training, which is formed by splitting the entire graph or inducing nodes (edges). Summarising, all sampling algorithms choose partial nodes based on specified rules and combine them into a single training batch(16).

1.6 GNNs Training Challenges

Unlike regular data, graph data consists of interconnected nodes and axes that translate to interconnected and inter-dependent vertices. GNN models each layer as a message-passing process in which each vertex accumulates the characteristics of its neighbors(18).This requires a more distinguished training approach than the regular DNNs and the complexity worseness when we scale the existing solutions. Practices like partitioning result in considerable communication costs since certain nearby vertices are necessarily separated between partitions(1).

A GNN model is commonly trained end-to-end once per task on the input graph using supervised data. Having adequate and diverse sets of labeled data to train specialized GNNs matching to each job is necessary for different tasks on the same graph. Access to adequate labeled data for those activities is usually prohibitively expensive and often impossible, especially for large-scale graphs(14).

1.7 Application of GNNs

We can find graphs in all the domain of life. One of the interesting topic to look at is chemistry. Indeed, the molecules have an inherent graph like structure. We can describe a molecule from the atoms being the nodes and the bonds between them the edges of the graph. GNNs have allowed us to learn more about existing molecular structures and also to discover new chemical structures. It has lead to great discoveries in computer aided drug design. One example is the prediction of the properties of new molecules by learning from existing one. We can also find GNNs in Natural Language Processing. Indeed, there exist way of converting a text into a graph of words from which we can use a GNN to obtain interesting results such as reading comprehension. However, one of the main application of GNNs today is e-commerce. Indeed, a lot of online platforms recommend content likely to interest a person who's browsing. The GNN will predict the most relevant content to show to the customer based on its information.

2 Research Question

The scope of this project is to conduct a comparative study between three state-of-the-art methods on GNNs namely DistDGL, Aligraph, and GraphTheta. We offer an overview of their training strategy, system architecture, and overall structural differences. Finally, we include a critical evaluation of the three methods concerning their efficiency and scalability in huge datasets that simulate real-world applications.

3 AliGraph

AliGraph (19)is a comprehensive graph neural network system with the goal of facilitating GNN training and enabling the development of new GNN algorithms by providing efficient graph storage and processing capabilities. AliGraph is now being used by Alibaba to support a range of business scenarios, including product suggestion and tailored search on the company's E-Commerce platform. AliGraph is made up of a distributed graph storage system, optimized sampling operators and runtime system. The system is designed to to supports existing popular GNNs as well as a series of developed ones at Alibaba.

3.1 System

The AliGraph platform consists of 5 different layers: application, algorithm, operator, sampling and storage where the three last layers compose the system.

The system architecture is abstracted from the generals GNN methods and is based on a GNN framework. This framework will take diverse information in input (such as a graph, vertex information etc...) and it will return an embedding for each vertex of the graph. These result in a system divided in three different layers. The storage layer that will fulfill the fast data access required by high-level operations and algorithms. In addition, we can find multiple operators that play an important role in GNN algorithms such as sample, aggregate and combine. Each of these operators reads data and complete distributed calculations. As sampling lays foundation to the other two, it results in the implementation of a sampling layer that have access the storage to generate training samples. On

Figure 2: Architecture of the AliGraph Platform

top of this layer, we can find the operator layer which aim to optimizes the aggregate and combine operators.

3.1.1 Storage

This layer aim to solve the challenges encountered by the large size of graph and provide efficient access in a distributed environment of clusters. Three different strategies are applied.

Graph Partition Because the platform is build on a distributed environment the whole graph is divided and stored separately in different worker nodes. This strategy aim to minimize the number of crossing edges whose endpoints are in different workers. To achieve this, the system use 4 different graph-partition algorithms (METIS(20), vertex cut and edge cut partitioning(21), 2-D partition(22), streaming-style partition(23)). As all these strategies are suitable to different circumstances, it is up to the user to chose which one he wants to use.

Separate Storage of Attributes The structural and the attributes of the graph are stored separately. The two main reasons for that are: attributes cost more space to store and attributes among vertices and edges overlaps. Thus, the amount of data stored is greatly reduced. However, the access time to retrieve the attributes is increased. The solution proposed consists to cache the frequently accessed nodes.

Caching Neighbors of Important Vertices This method consists to locally cache the neighbors of the important vertices in each worker memory to reduce the communication cost. Indeed, if a vertex is frequently accessed, storing its neighbors will reduce the visiting cost of other vertices to their neighbors. However, it comes with the price of space if there are too many neighbors. In order to make the best trade-off between storing and communication, the system uses a metric that will give a degree of importance to each vertex. Later, only the vertices that score higher than a manually selected threshold will be stored.

3.1.2 Sampling

To build embeddings for each vertex, GNN algorithms rely on pooling neighborhood information. However, the degree of skewed distribution of graphs, makes the convolution procedure difficult to perform. Thus, a sample layer is abstracted due to its relevance in the system. The AliGraph system will abstract three kinds of different samplers:

- Traverse: it will be used to sampling a batch of vertices or edges from the whole partitioned subgraphs.
- Neighborhood: it will be used to generate the context for a vertex.

- Negative it is used to generate negative samples to accelerate the convergence of the training process

According to the neighborhood requirements, the graph is partitioned by source vertices. The vertices are split on a graph server by group. Each group will be related with a request-flow bucket where the operations are all about the vertices in the group. The bucket is a lock-free queue and each operation in the bucket will be executed sequentially leading to enhanced performances from the system.

3.1.3 Operator

After the sampling, the output data can be processed easily. The system use two kinds of operators that will use the data.

- Aggregate: it collects the information from the surrounding neighborhood of a node and could be compared to a convolution operation.
- Combine: it allows to describe a vertex using the information of its neighbors.

In addition, new operators can be defined. The operators will be distributed operators and the data required for calculation will be distributed on each Server of the service.

3.2 Design of Algorithm

On top of the system, we can find the algorithm layer where GNNs can be built on AliGraph system. Indeed, the AliGraph platform is abstracted from the GNN algorithms. Thus, a large number of GNNs can be build on the platform. In order to implement a GNN algorithm, the algorithm needs to use the AliGraph's sampling operator instead of its own. In addition, it needs to instantiate the Agregate and Combine operators by adapting the algorithm. There are already multiple GNN algorithms that are native with the platform. After what, the algorithm can be trained normally.

4 DistDGL

In most applications that use GNN the main problem they face is that the graphs are large(24). That created the need for a new method that would be able to solve this issue. That method is DistDGL and it is based on the Deep Graph Library (DGL). It spreads GNN model mini-batch training over a cluster of processors. The data dependency between vertices distinguishes GNN mini-batch training from other neural networks. As a result, we must select subgraphs that capture the data with care.

4.1 Graph Partition

The purpose of graph partitioning is to divide the input graph into numerous divisions with the fewest possible edges between them. Before distributed training, graph partitioning is a preprocessing procedure. The overhead of a graph is reduced by partitioning it once and using it for numerous distributed training runs.

The algorithm used in DistDGL's graph partitioning is METIScite(20). DistDGL adjusts that algorithm so that the neighbors of the local vertices are accessible on the partition, allowing samplers to compute locally without having to communicate with one another.

4.2 System

DistDGL consists of the following logical components: sampler, KVStore server for storage, trainers and a model update.

To decrease network communication, the general principle is to delegate computing to the data owner. The input graph is initially partitioned by DistDGL using a light-weight min-cut graph partitioning technique. The edge features are then partitioned and co-located with graph partitions. To supply the local partition data, DistDGL starts the sampler and KVStore servers on each machine (Figure 2).

For each trainer (their use is explained later), DistDGL can spawn multiple sampling worker processes to sample mini-batches in parallel. Instead of sending distant requests, the workers use shared memory to access the graph structure stored on the local sampler server.

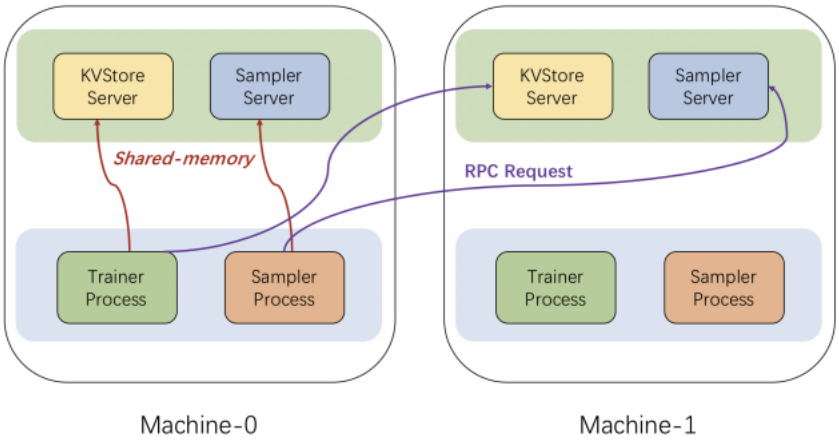


Figure 3: Architecture of the DistDGL System

4.3 Sampler

The sampling of the mini-batch graph derives, as expected, from the input graph. DistDGL does neighbor sampling to reduce computation. The trainers receive the mini-batch graphs created by those samplers.

DGL has created a set of adaptable Python APIs to enable a wide range of sampling methods that have been proposed in the literature. This API architecture is preserved in DistDGL, although it is implemented differently internally. The trainer sends out sampling requests using the current mini-target batch's vertices at the beginning of each iteration. The graph partitioning technique generates a core vertex assignment, which is used to dispatch the requests to the machines. When sampler servers get a request, they contact DGL's sampling operators on the local partition and return the results to the trainer process. Finally, the trainer compiles the results into a mini-batch.

4.4 KVStore

The KVStore in general stores your data as key-value pairs in collections. In our case, it has two easy-to-use interfaces for getting data from or sending data to the distributed storage. It also controls the vertex embeddings if the user-defined GNN model specifies them.

To manage the vertex and edge features, as well as vertex embeddings, DistDGL creates a distributed in memory KVStore. Flexible partition policies to map data to separate machines are supported by DistDGL's KVStore. Each machine's graph partitions are aligned with DistDGL's different partition policies for vertex and edge data. Because the majority of transmission in GNN distributed training is spent accessing vertex and edge features, efficient data access in KVStore is critical. Using shared memory is a fundamental enhancement for rapid data access. Because data and processing are co-located, most data access to KVStore is handled by the KVStore server on the local workstation. Trainers can gain immediate access to the majority of the data without incurring any communication or process/thread scheduling costs. For fast networks, the RPC frameworks are used to optimize network transmission of DistDGL's KVStore . DistDGL's KVStore is designed to provide sparse embedding for training transductive models with learnable vertex embeddings in addition to storing feature data. The embedding is updated by KVStore based on the optimizer that the user has registered.

4.5 Trainers

The trainers acquire the information of the mini-batch graph from the samplers and the corresponding collection features, in this cases the edges of the graph, from the KVStore. The gradients of dense parameters are provided to the dense model update component for synchronization, which is the next component we will explore, whereas the gradients of sparse embeddings are returned to the KVStore.

DistDGL uses a two-level method to partition the training set equitably across all trainers at the start of distributed training to balance the computation in each trainer. To begin, each trainer has to have the same quantity of training examples. Training samples are evenly split by IDs and are assigned the ID range to the machine with the biggest graph partition overlap with the ID range. This is achievable because during graph partitioning, vertex and edge IDs are relabeled , and the vertices and edges in a partition have a contiguous ID range. There is a choice between load balancing and data localization. In reality, the tradeoff is insignificant as long as the graph partition algorithm balances the number of training samples between partitions. It's discovered that a random split results in a very balanced workload assignment. The updating of the parameters in terms of synchronization, is done through synchronous SGD(25). To overlap communication and computation, asynchronous SGD(26) is employed to update the sparse vertex embeddings.

4.6 Dense Model Update

This component is used for aggregating dense GNN parameters to perform synchronous SGD(25) which is an optimization algorithm that can be used to train neural network models.

5 GraphTheta

GraphTheta is a new parallel and distributed graph learning system that supports a variety of training strategies and allows for efficient and scalable learning on large graphs. GraphTheta's three main goals are to support dense and (highly) skewed sparse graphs, to investigate new training strategies in addition to the existing mini-batch and global-batch methods, and to provide deep neighborhood exploration without neighbor sampling. GraphTheta implements both localized and globalized graph convolutions on graphs, and implements NN-TGAR, a new graph learning abstraction which is intended to bridge the gap between graph processing and graph learning frameworks. Furthermore, a distributed graph engine is used to carry out the stochastic gradient descent optimization with hybrid-parallel execution(27).

5.1 System

GraphTheta was inspired by distributed graph processing systems and can handle three training techniques at the same time: NN-TGAR, Forward and Auto-diff, and Backward. It balances memory consumption, time cost per epoch and fast convergence to enable deep GNN exploration without pruning graphs.

It consists of five parts: i) a graph storage component with distributed partitioning and heterogeneous feature and attribute management, (ii) a subgraph generation component with sampling techniques, (iii) graph operators that modify nodes and edges, and (iv) learning core operations such as neural network operators (containing fully-connected layer, attention layer, batch normalization, concat, mean/attention pooling layers), usual loss functions like softmax cross-entropy loss and optimizers. The system components of GraphTheta are shown in Figure 4.

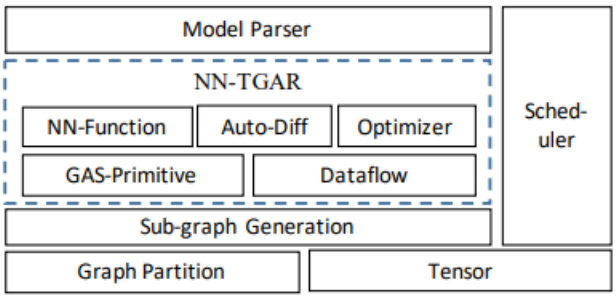


Figure 4: Architecture of the GraphTheta System

5.2 Graph Partition

The core graphs of GraphTheta are put in a distributed setting, that requires effective graph partitioning techniques.A number of graph partitioning approaches have been implemened, such as vertex-cut, edge-cut, and hybrid-cut solutions. GrapthTheta, also includes popular graph partitioning algorithms that allow for many popular graph processing methods.

A novel graph partitioning approach is used to evenly distribute nodes to partitions and cut off cross-partition edges. Master and mirror nodes are used, with a master node allocated to one partition and mirrors established in other partitions. Our strategy places each edge in the partition in which its source node is a master (target nodes also can be used as the indicator). This ensures that every edge has at least one master node.

To avoid the problem of duplicated mirror nodes from the vertex-cut techniques, that previous systems have encountered(28), GraphTheta allows mirror nodes to function as placeholders, keeping just the node states rather than the real data.

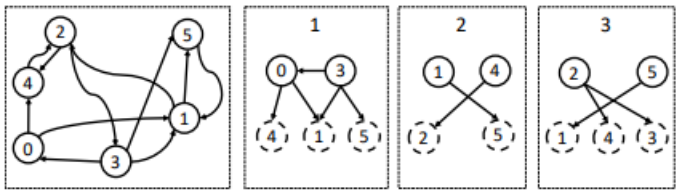


Figure 5: GraphTheta. Three partitions with evenly assigned nodes

5.3 Graph Traversing and Storage

GraphTheta keeps node and edge data individually and arranges outgoing edges in Compressed Sparse Row (CSR) and incoming edges in Compressed Sparse Column (CSC). The distributed graph traversal is performed in two parallel operations: one with CSR and the other with CSC. For the CSR, each master node transmits its associated values to all mirrors and then gathers its outgoing edges with master neighbors, skipping the edges with mirror neighbors. Each mirror node aggregates incoming edges with master neighbors for the CSC procedure. Mirror nodes that get values from their masters will be passively gathered by their neighbors.

The technique used can handle the issue of local message bombing. We only need one message propagation of node values and outcomes for a master-mirror pair, which decreases the traffic burden from $O(M)$ to $O(N)$. During the dataflow execution of a GNN model, however, we only synchronize node values involved in the computation per neural network layer.To adjust cluster-batched training, heuristic graph partitioning methods such as METIS(20), and Louvain(29) are also supported.

5.4 Sampling

GraphTheta implements a new type of training strategy called cluster-batched training, which conducts graph convolution on a cluster of nodes and may be thought of as an extension of existing mini- or global-batch strategies to reduce unnecessary calculations between batches(13).

Cluster-batch divides a large graph into smaller clusters first. Then, either on one cluster or a mix of clusters, it creates a batch of data. Cluster-batch, like mini-batch, conducts localized graph convolutions. Moreover, cluster-batch condenses the neighbors of a target node into a single cluster, which is the same as performing a generalized convolution on a cluster of nodes. It typically creates clusters by employing a community discovery technique that maximizes intra-community edges while limiting inter-community connections(29). Community detection can be performed either offline or in real-time, tailored to the needs of the system. Furthermore, cluster sizes are frequently uneven, resulting in a wide range of batch sizes.

5.5 Training

5.5.1 Subgraph Training

After the sampling methods are applied, naturally we proceed to the subgraph training. In the paper, it was given an example of mini-batch training which is indicative of the training process Figure 6. The right side depicts a subgraph built from a set of initial target nodes 1,2,3 with one-hop neighbors 4,5,6 and two-hop neighbors 7,8. The forward and backward computation of this subgraph is shown on the left, with arrows showing the propagation direction.

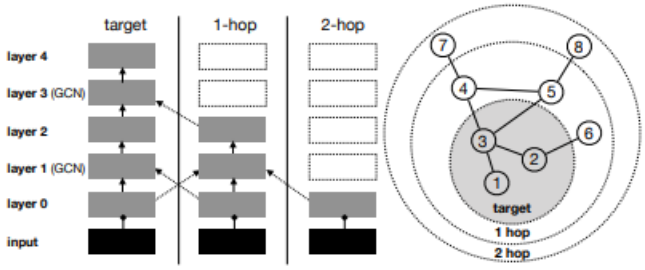


Figure 6: Target node tensor stacks and its 2-hop neighbors in the training process with a 2-hop GNN model

Computational savings A simple way of computing subgraphs is to load the subgraph structure and relevant data into memory and then conduct matrix operations on the subgraph on the same processor. This approach has an inherent constraint in generalization since the memory overhead of a subgraph may surpass the memory limit. GraphTheta uses distributed computing to finish the training method by simply traversing the distributed subgraph's structure. It uses a breadth-first search traverse operation to build a subgraph. This action initializes a minimum number of layers per target node that are engaged in the computation in order to prevent excessive graph computing propagation.Furthermore, within each process/worker, a vertex-ID mapping is developed between the subgraph and the local graph to reuse CSR/CSC indexing, avoiding the expense of subgraph structure building and preserving graph access efficiency.

5.5.2 Concurrent train of multiple subgraphs

The power of a distributed system cannot be fully redeemed by sequentially training subgraphs. GraphTheta was created to train numerous subgraphs with multi-versioned parameters in a distributed environment while also allowing for concurrent lookups and modifications. As a result, the present GNN training methods have two distinct features: i) parallel subgraph tensor storage based on distributed graphs, and ii) GraphView abstraction and multi-versioned parameter management for parallelized batched training. The topology of the parallel batched training that is used for GraphTheta is depicted in Figure 7

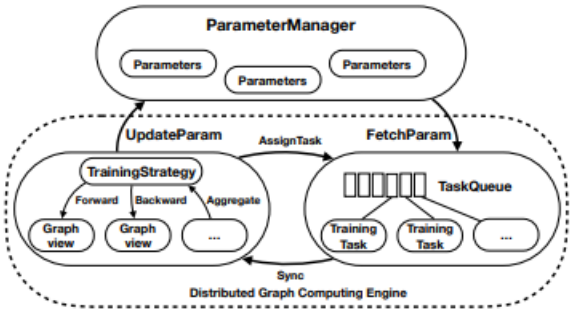


Figure 7: Parallel batched training - GraphTheta

Parallel tensors storage Two essential strategies are inspected for subgraph parallel training to enable low-latency access to dispersed subgraphs and minimize total memory overhead. Reusing the CSR/CSC indexing is the first approach. Constructing and releasing indexing data for each subgraph on the fly is wasteful in a high-concurrency situation. Instead, as mentioned in section Subgraph Training, the global indexing of the whole graph is reused, and a private cache-friendly vertex-ID mapping is employed to efficiently retrieve a graph topology.

Task-oriented tensor storage is the second method. In GNNs, the same node, especially high degrees nodes, might appear in many subgraphs generated from separate batches of target nodes. The memory structure of nodes is task-specific (a task might be an individual forward, backward, or aggregate phase) and sliced into frames to make tasks context independent and hide underlying implementation details. A frame is a stack of sequential resident memory that stores raw data and tensors for a specific node. Memory can be dynamically allocated and freed every frame on the fly to lessen peak memory load. The output tensors for each layer are computed and released immediately after usage in the forward/backward phase.

Because tensor allocation and de-allocation have a context-aware memory consumption pattern, a tensor caching system is built between frames and normal memory manipulation libraries to minimize frequent trapping in operating system kernel areas.

GraphView was created to train numerous subgraphs at the same time. All major characteristics of the underlying parallel graph storage, such as reused indexing, embedding lookup, and the distributed graph representation, are maintained by it. The GraphView, which is implemented as a lightweight logic view of the global graph, exposes a set of interfaces that are required by all training strategies and enables easy communication with storages.Furthermore, other training procedures, in addition to global-, mini-, and cluster-batch, may be implemented using GraphView. Training tasks are also scheduled in parallel, allowing a training worker to be assigned to the different forward, backward, and aggregation stages at the same time. To increase load balancing and efficiency, a work-stealing scheduling method is used due to the diverse workloads of subgraphs.

6 Discussion

We divided the methods into three categories to have a better overview of the techniques used in each of the approaches. We decided to divide them such as we have: the partitioning, the sampling and the training. In Table 1 we have made a summary table of these categories. First of all, we look at graph partitioning. DistDGL use the METIS algorithm. Metis can also be used by AliGraph but Aligraph supports more partitioning algorithms. On the other hand, GraphTheta uses its own graph partitioning algorithm, as well as METIS. In addition, the sampling in DistDGL is a mini-batch sampling with neighbours that could be compared to the neighbourhood sampling of AliGraph. Contrarily to the two previous approaches, GraphTheta uses a new type of sampling which is cluster batch sampling, however, it can also use the conventional ones such as mini-batches sampling. The distributed operators utilized by the algorithms on the platform enable distributed training in AliGraph. However, it is offered the possibility to utilize your own GNN algorithm, as long as you modify it such that it is complainant to the AliGraph operators. By using data-parallel execution, the training will be performed using forward and backward computation over a mini-batch by only one worker. Trainers in DistDGL perform backward and forward computations in parallel on their own-mini batches. Finally, GraphTheta employs parallel tensor storage as well as GraphView.

Based on the above mentioned observations, the techniques used by AliGraph are similar to those used by DistDGL, but they also support new ones that DistDGL does not support. GraphTheta, on the other hand, appears to be the most efficient and scalable of the methods due to its novel implementation of graph partitioning and unique sampling technique. GraphTheta distinguishes itself from other approaches by utilizing hybrid-parallel execution, which computes each mini-batch by a group of processes/workers and supports several training strategies such as global-batch, mini-batch, and cluster-batch.

In a distributed Linux CPU Docker environment, it scales approximately linearly using up to 1,024 workers. Given the cheap cost of access to a cluster of CPU machines in public clouds such as Amazon Web Services, Google Cloud, and Microsoft Azure, GraphTheta has the potential to enable low-cost graph learning on large graphs.

Method	Partitioning	Sampling	Training
AliGraph	The algorithm can be chosen in function of the graph data: METIS, Vertex cut and edge cut partitioning, 2D-partitioning, Streaming style partitioning.	Traverse, Neighborhood, Negative	The GNN algorithms need to be adapted to the platform. They need to implement the sampling as well as the operators defined by the aligraph system. These operators run in a distributed way on different servers allowing faster training.
DistDGL	METIS: the neighbours of the local vertices are accessible on the partition. Samplers work locally without interference from the other partitions	Mini-batch sampling with neighbours	Training samples are uniformly divided by IDs. The machine that has the largest graph partition that overlaps with the ID range is assigned the ID range. The trainers compute the gradient by running the forward and backward computations in parallel on their own mini-batches.
GraphTheta	Specific to GraphTheta: evenly distribute nodes to partitions and cuts off cross-partition edges. The master node is allocated to one partition and mirrored in other partitions. It avoids duplicated mirror nodes, the mirrored nodes can function as placeholders keeping the state of the node and not the real data.	sampling without neighbours cluster batch	Parallel subgraph tensor storage. Within each process, a vertex id mapping is developed between the subgraph and the local graph to reuse CSR/CSC indexing. Task-oriented tensor storage: the same node can be incorporated in different subgraphs. GraphView provides a set of interfaces necessary for all training strategies. The training tasks are scheduled in parallel.

Table 1: Summary table of each method

When compared to other state-of-the-art methods, including DGL and Aligraph, GraphTheta appears to have the best generalization potential. The novel cluster-batch training technique that GraphTheta uses, achieves the greatest generalization results and the fastest convergence time on massive networks, according to experimental results on the Alipay dataset, the biggest edge-attributed graph ever used to evaluate deep GNN models in the literature. It is worth noting, however, that while the system may handle a wide range of activities, only node classification tasks were examined, hence the performance of GraphTheta in comparison to other jobs is unclear. Concluding, if we had to choose an approach for our own implementation, we would go with GraphTheta, then DistDGL, and finally AliGraph.

7 Conclusion & Future work

The use of graph data has exploded during the last years. The graph neural network algorithms offer a wide range of new applications in every domain. However, their training is still an important challenge due to the large size of the graph data/ In this study, we talk about three different methods for training Graph Neural Networks in a distributed way. We present the methods as well as the strategies that they use to make the training of the GNN algorithms faster. Finally, we divide each method into three parts and compare the techniques used in each one of them. Because Graph neural networks are among the most powerful tools in deep learning the opportunities for research are very large.

Indeed, as we describe it in our study, the approaches focus especially on the training but there is also a large room for improvement in the development of efficient GNN inference. We could see some improvement concerning advanced mini-batching or cluster-batching in GNNs where the systems would use asynchronous or bi-directional mechanisms making it more efficient. Finally, we observe

that methods rely on the use of graph partitioning but replication could also be used to increase the overall performance.

References

[1] M. Serafini and H. Guan, "Scalable graph neural network training: The case for sampling," *ACM SIGOPS Operating Systems Review*, vol. 55, no. 1, pp. 68–76, 2021.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[3] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[5] Z. Song, X. Yang, Z. Xu, and I. King, "Graph-based semi-supervised learning: A comprehensive review," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[6] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 891–900.

[7] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[9] H. Attiya and J. Welch, *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004, vol. 19.

[10] R. Liao, M. Brockschmidt, D. Tarlow, A. L. Gaunt, R. Urtasun, and R. Zemel, "Graph partition neural networks for semi-supervised classification," *arXiv preprint arXiv:1803.06272*, 2018.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[12] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.

[13] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.

[14] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1857–1867.

[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[16] X. Liu, M. Yan, S. Song, Z. Lv, W. Li, G. Sun, X. Ye, and D. Fan, "Gnnsampler: Bridging the gap between sampling algorithms of gnn and hardware," *arXiv preprint arXiv:2108.11571*, 2021.

[17] X. Liu, M. Yan, L. Deng, G. Li, X. Ye, and D. Fan, "Sampling methods for efficient training of graph convolutional networks: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 2, pp. 205–234, 2021.

[18] M. Y. Wang, "Deep graph library: Towards efficient and scalable deep learning on graphs," in *ICLR workshop on representation learning on graphs and manifolds*, 2019.

[19] R. Zhu, K. Zhao, H. Yang, W. Lin, C. Zhou, B. Ai, Y. Li, and J. Zhou, "Aligraph: A comprehensive graph neural network platform," 2019. [Online]. Available: https://arxiv.org/abs/1902.08730

[20] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[21] F. Rahimian, A. H. Payberah, S. Girdzijauskas, and S. Haridi, "Distributed vertex-cut partitioning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2014, pp. 186–200.

[22] E. G. Boman, K. D. Devine, and S. Rajamanickam, "Scalable matrix computations on large scale-free graphs using 2d graph partitioning," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 1–12.

[23] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1222–1230.

[24] D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang, and G. Karypis, "Distdgl: distributed graph neural network training for billion-scale graphs," in *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. IEEE, 2020, pp. 36–44.

[25] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," vol. 3, 2017.

[26] Q. Meng, W. Chen, J. Yu, T. Wang, Z.-M. Ma, and T.-Y. Liu, "Asynchronous accelerated stochastic gradient descent," *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016.

[27] H. Li, Y. Liu, Y. Li, B. Huang, P. Zhang, G. Zhang, X. Zeng, K. Deng, W. Chen, and C. He, "Graph-theta: A distributed graph neural network learning system with flexible training strategy," *arXiv preprint arXiv:2104.10569*, 2021.

[28] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "{PowerGraph}: Distributed {Graph-Parallel} computation on natural graphs," in *10th USENIX symposium on operating systems design and implementation (OSDI 12)*, 2012, pp. 17–30.

[29] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

Web Services and Cloud-Based Systems Literature Study IDaaS (Identity as a Service) - Research on Privacy Enhancement Methods

ChiaYu Lin
13692577
[1], Wenjun Liang
14128012
[1], and Jianyang Gu
13526731[1]

[1]Department of Computer Science, Universiteit van Amsterdam, Amsterdam, the Netherlands

Abstract

This literature review explores Identity as a Service(IDaaS) and further focuses on some privacy enhancement methods. IDaaS is an on-trend cloud-based identity management service while facing some privacy issues. In this paper, we analyzed three privacy enhancement methods (Rupa, Kuhrmann, and Gomaa) and compare these methods. We discovered that SAML is the most ideal identity authentication protocol used in IDaaS environment. Furthermore, among three studies, Rupa and Gomaa's method showed a low data transmission latency; while Fuhrmann's method showed an efficient performance over large data size processing.

1 Introduction

1.1 Background

To get a comprehensive understanding of Identity as a Service (IDaaS), we first look into Identity Access and Management(IAM) and cloud identity management. With the development of cloud computing and a burst of cloud services, digital identity in the cloud should be well managed. Digital identity is a series of characteristics that an entity owns and uses information technologies to define an identity (a person, company, application, or device)(6). However, digital identity management is not a simple process, and cloud identity management solutions need the implementation to protect digital identity. We may assume Identity and Access Management as a way of saving our passwords, while it is way more this.

Identity Access and Management (IAM) is a traditional security technology that ensures users with different roles have different privilege access to resources. Identity and access management is one of the staples of a comprehensive cybersecurity platform—possibly the most important if some surveys are to be believed(7).

In the modern business context, cloud identity management can be seen as the next step or next generation of identity and access management solutions(7). There are two situations for cloud identity management: private cloud model and public cloud model. Typically, it is much easier to manage identity in a private cloud model because you own and manage the systems and applications, any customization that you need to make to integrate the cloud service into your current identity management scheme can be performed during the design and deployment stage of cloud services(8).

Group 23 Literature Study of Web Service and Cloud-based System (2022), Amsterdam, the Netherlands.

For the public cloud model in cloud identity management, IDaaS is one of the most important models in use.

Identity as a Service (IDaaS) has many variations, but most of them are just a combination of each other. 'As a service' refers to the way IT assets are consumed in these offerings - and to the essential difference between cloud computing and traditional IT. In traditional IT, an organization consumes IT assets - hardware, system software, development tools, and applications - by purchasing them, installing them, managing them, and maintaining them in its own on-premises data center(9). The cloud service will provide the customers with managing and maintaining the assets, and the customer would consume via an Internet connection, and pays according to the cloud service subscription.

With many types of cloud services offering, it is meaningful to review some other could service models first. IaaS, PaaS and SaaS are the three most popular types of cloud service offerings(9).

- IaaS(infrastructure as a service), is on-demand access to serve storage and network. With the cloud-based physical and virtual servers, storage, and networking - the backend IT infrastructure for running applications and workloads in the cloud(9).
- PaaS(platform as a service), is access to OS and a complete, ready-to-use, cloud-hosted platform for applications.
- SaaS(software as a service), is access to packaged software OS and ready-to-use, cloud-hosted application software.

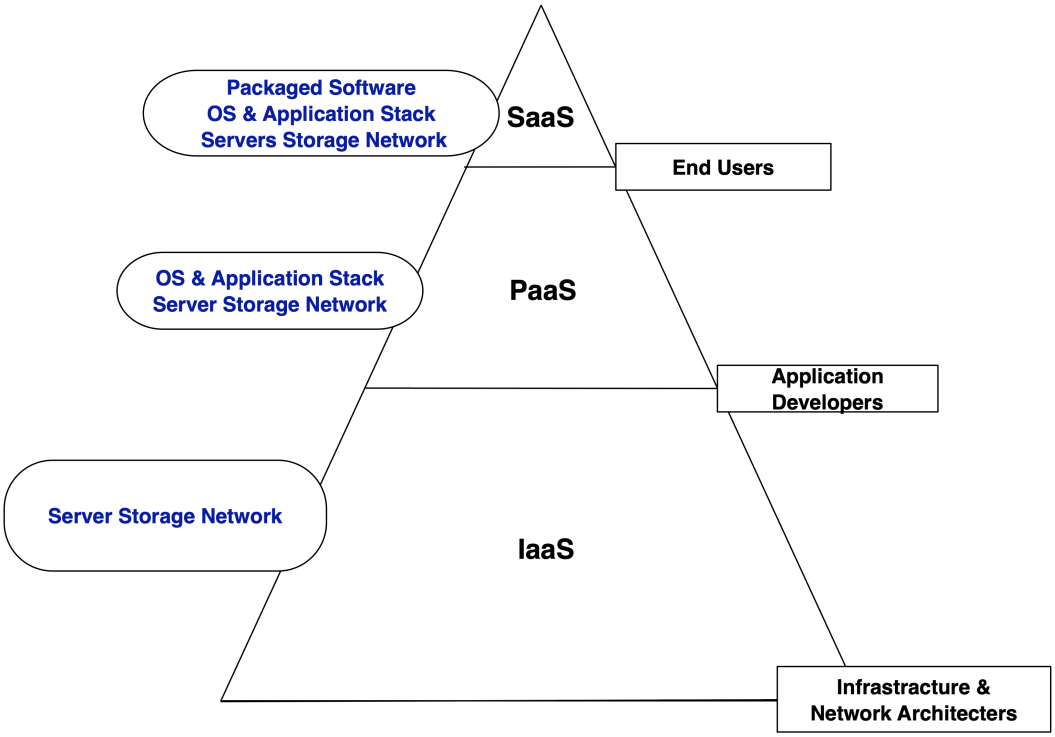


Figure 1: Cloud Service Models (IaaS,PaaS, and SaaS)

Identity as a Service is a variation of these services above using a third party. As the architecture shown in Fig2, administrators create an account at the IDaaS provider (directly through the UI or programmatically through an API), and the provider will then create accounts in the cloud service applications used by your company, leveraging federation when possible(17). IDaaS is flexible and can provide many on-ramps to different cloud services. Moreover, it can help with the identity service integration, just as how its name stated.

We have to point out, however, in IDaaS model the actual accounts and credentials could be stored in the enterprise identity management system, the IDaaS identity management system, or the cloud

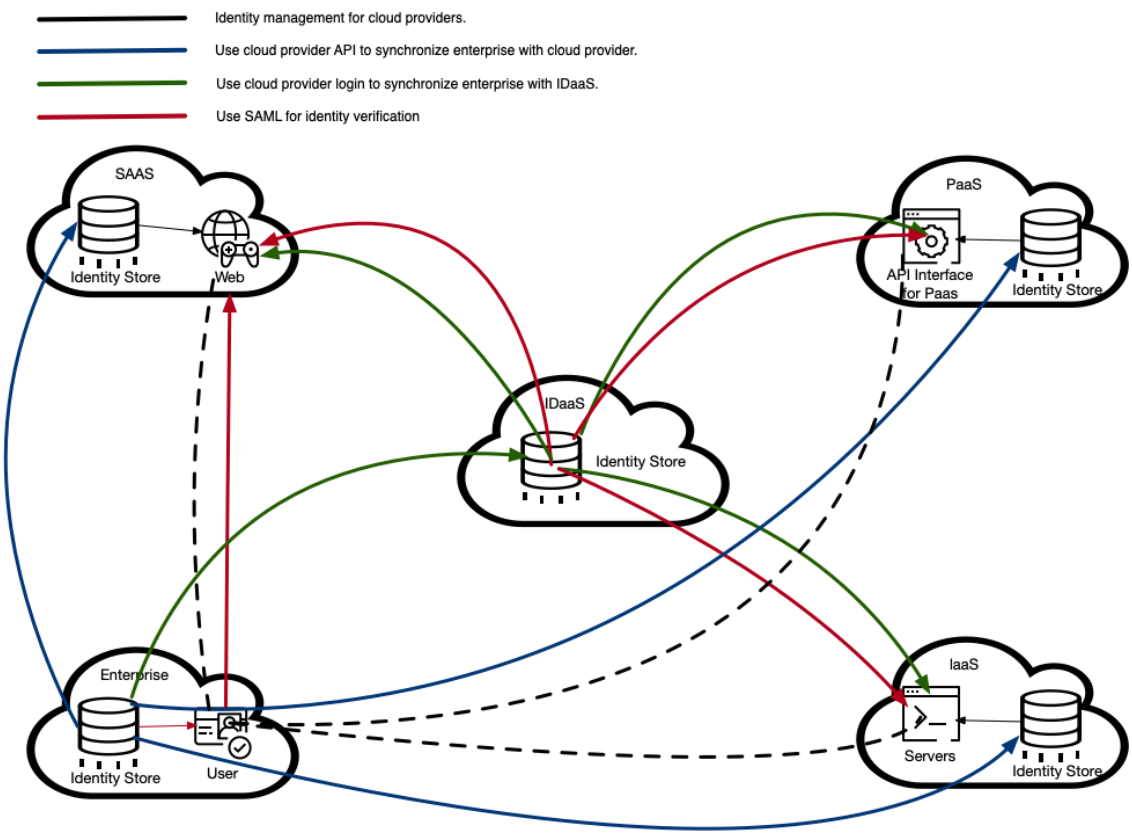


Figure 2: Identity Flows and Stores in the Cloud

service identity management system(17). As a result, The main disadvantage of this model is the privacy issue: security is always treated as the main concern for companies and users from moving to the cloud. Also, it has the disadvantage with any cloud-based service: all the applications and users would go down if the system breaks down. Our paper would focus on the privacy issue of IDaaS and dive deep into three strategies for improving IDaaS privacy.

1.2 Traditional IAM to Cloud IDaaS

Traditional IAM includes an internal identity authentication service within the corporation to verify user identities. This causes a heavy workload on the system to perform each identity verification process. Furthermore, users are registered inside the corporation, therefore, all user identities are maintained in a single large internal database, which demands ongoing maintenance and policy to keep data secured.

IDaaS is a similar technology to traditional IAM, the difference is, IDaaS deployed its identity services on the cloud. By using IDaaS, the corporation does not require any authenticate mechanism of its own to verify user identity. Instead, the process of user identification is provided by a trusted Identity Provider(IdP).(5) Besides, IDaaS provides better business services such as tracking user transactions, easy to modify user privileges, user behaviour analysis. Comparing to traditional IAM where users are registered within the corporation and user information is stored in a local database, IDaaS requires users to register themselves on the cloud, and identities are passed between IdP and Service Provider(SP) to grant permission to resources. This may lead to privacy concerns about user data which would be discussed in the next section. Table 1 is a comparison of traditional IAM and IDaaS.

Table 1: Comparison of Tradition IAM and IDaaS

	Traditional IAM	**IDaaS**
Deploy	Within the corporation.	On public or private clouds.
Authenticate	Authentication is performed using an internal authentication service.	User identification is provided by a trusted IdP.
Other functions	Need to deploy and maintain by the corporation itself.	Uses existing service provided by cloud provider.
Performance	Depends on the corporation's server configuration, maybe cause heavy workload to system.	Multiple cloud server operating at the same time.
Privacy	User data usually stored in a local database, low risk of data leak (excluding human errors)	User identities are exchanged between the IdP and the SP which leads to high risk of data leak.

1.3 Challenges

Among cloud service models such as SaaS, PaaS, and IaaS, IDaaS is a relatively new and immature technology that still contains multiple hidden security risks to be resolved. Before adopting IDaaS from traditional IAM, the corporation needs to perform a risk assessment for all of its applications that might use IDaaS authentication services and consider the potential security risks of it(10). In this section, the three major challenges of IdaaS - *availability, identity data protection, and trusting a third party*(11); would be discussed.

Identity data protection

In IDaaS cloud environment, user identities are constantly exchanged between users, IdPs, and SPs, how to ensure identity data protection is arguably the biggest challenge for IDaaS. One can enhance the privacy of identity by adopting the CIA (Confidentiality, Integrity, and Availability) security triad(12). For data confidentiality, IDaaS must ensure that the identity is only disclosed to authorized users, which necessitates the use of encryption techniques. For ensuring data integrity, it becomes more complex since the cloud environment contains multiple databases, and identities can easily get tampered with or lost during data exchange. IDaaS must apply a unique access management technique (14) to prevent this situation.

Availability

The availability of IDaaS, which is related to whether the identity service could operate normally by handling such a large amount of identity exchange, authentication, and authorization, is a critical issue given the rapid growth of digital identity requests(13). To ensure the satisfactory performance of IDaaS, how to achieve its cloud scalability while remaining low-cost, and prevent cyber attacks that may cause service downtime and user unsatisfactory is one of the major challenges.

Trusting third party

In cloud environment, IdPs and SPs are considered as third parties that are responsible for identity authentication and authorization. This prompts the concern of whether users can trust these third parties. Therefore, the issue of how to apply techniques such as Blind Identity Management (BlindIdM), which allows third parties to provide identity services without knowing the users' actual identity information, is also a challenge for IDaaS(15).

1.4 Research Question

Due to the potential security risks in IDaaS cloud environment. Recent research into how to improve the privacy of IDaaS has been a popular issue. Studies have recommended various solutions for improving IDaaS's privacy problem. In this paper, **we will discuss and compare the performance and algorithm used by three privacy improvement strategies based on distinct authentication models.**

2 Literature survey

Privacy preservation is a significant issue for allowing consumers and businesses to believe that the data they share and communicate on the cloud is secure. Several strategies for providing various encryption protocol techniques with infrastructure-based services have been proposed. In this section, three methodologies for improving the privacy of IDaaS in the cloud environment will be introduced.

2.1 SAML and PECC based authentication protocol in fog computing

Rupa *et al.*(1) suggested a fog based SAML and PECC (Pentatope found Elliptic Curve Crypto Cipher) based authentication protocol model. This model improved user privacy by concealing data transferred over the cloud server. In this section, a brief overview of SAML and the architecture of this authentication based protocol model will be discussed.

2.1.1 Introduction of SAML

SAML(Security Assertion Markup Language) is a well-known identity protocol used in cloud infrastructure(3). SAML is an XML-based protocol that generates security tokens from end-users information, this token serves as an authentication symbol provided by the identity provider(IdP). By passing the authenticated token to the service provider(SP), end-users can grant permission to access the service(4).

2.1.2 Introduction of PECC

PECC (Pentatope found Elliptic Curve Crypto Cipher) is a key generation, encryption, and decryption technique used to secure data. PECC could help improve user privacy in Rupa's protocol model by hiding data information from identity providers (16).

2.1.3 Authentication protocol model

In Rupa's authentication protocol model, to join the data communication under this model, all clients and parties must first register with the authentication server (AS) to grant permission and obtain an encrypted authentication token. Step 1 to 3 and step 7 to 9 in Fig.3 shows the procedure of User A and User B sending a request to the *Authentication Server(AS)* to receive a token encrypted with an encryption key(AS-IDS), and a *timestamp T1*.

Based on the time stamp, this authentication token will validate the authenticity of clients at the Identity Server (IDS), and the clients will receive an authorized Ticket and a random key to communicate with the other user. Step 4 to 6 and 10 and step 12 to 9 in Fig.3 shows the procedure of User A and User B sending information includes encryption key(AS-IDS) which retrieves from the previous step and the User ID of A and B. User A and B will then receive a *random key RK(A-B)* for them to communicate along with a *Ticket* encrypted with *encryption key(IDS-A/IDS-B)*.

To initiate communication between two clients, the requester sends a PECC encrypted message to the receiver, which contains a random key that encrypted the original message as well as the receiver's secret key that encrypted its Ticket. The message could then be decrypted by the receiver using the random key shared by the two clients. Step 12 and 13 in Fig.3 shows the procedure of User A generating a communication channel to User B with the *RK(A-B)* which encrypted the message and the *encryption key(IDS-A)* which encrypted the ticket.

All communication in this model is session-based, with timestamps that expire after a set period of time. Following that, clients must reauthorize themselves with the AS, which reduces the likelihood of data theft attacks.

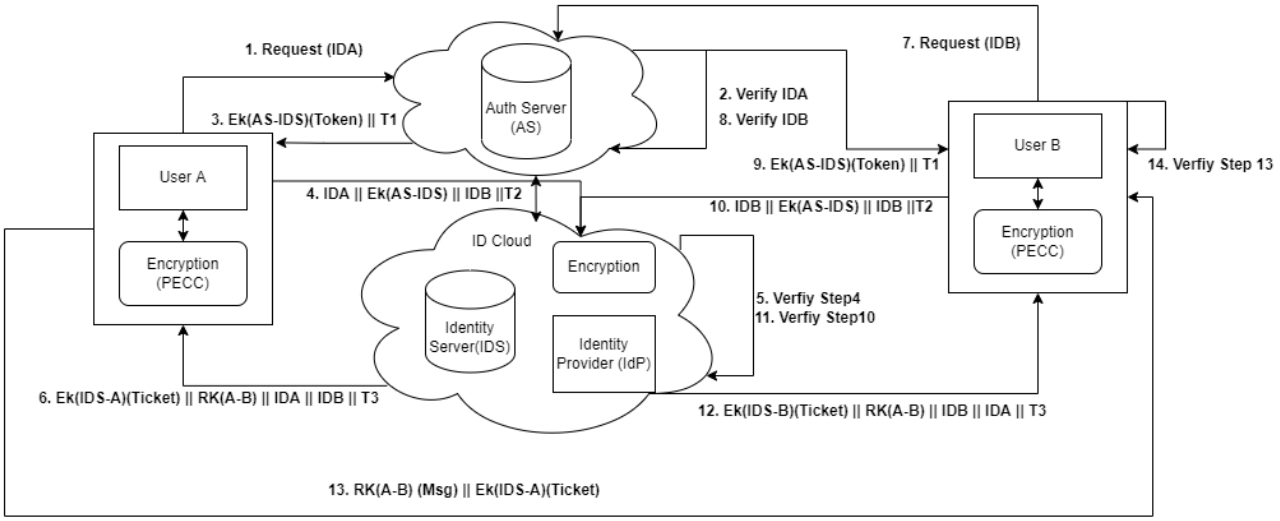


Figure 3: Protocol Model and Transaction

2.2 Trust Adaption and Purpose-based Encryption to protect the disclosure of PII(Personal Identifiable Information)

Fuhrmann *et al.*(2) proposed a Trust adaption model and Purpose-based Encryption to protect privacy. This approach is suitable for sharing sensitive personal data in a large, distributed, heterogeneous environment. Personal Identifiable Information customers stored in cloud environment may cause disclosure of PII from intermediary entities and from untrusted hosts. They proposed an efficient solution to protect the PII of customers over the intended channel and unintended channel. Also, their solution help against an untrusted host.

2.2.1 Architecture Design Overview

2.2.2 Design Principles for Trust Adaptation

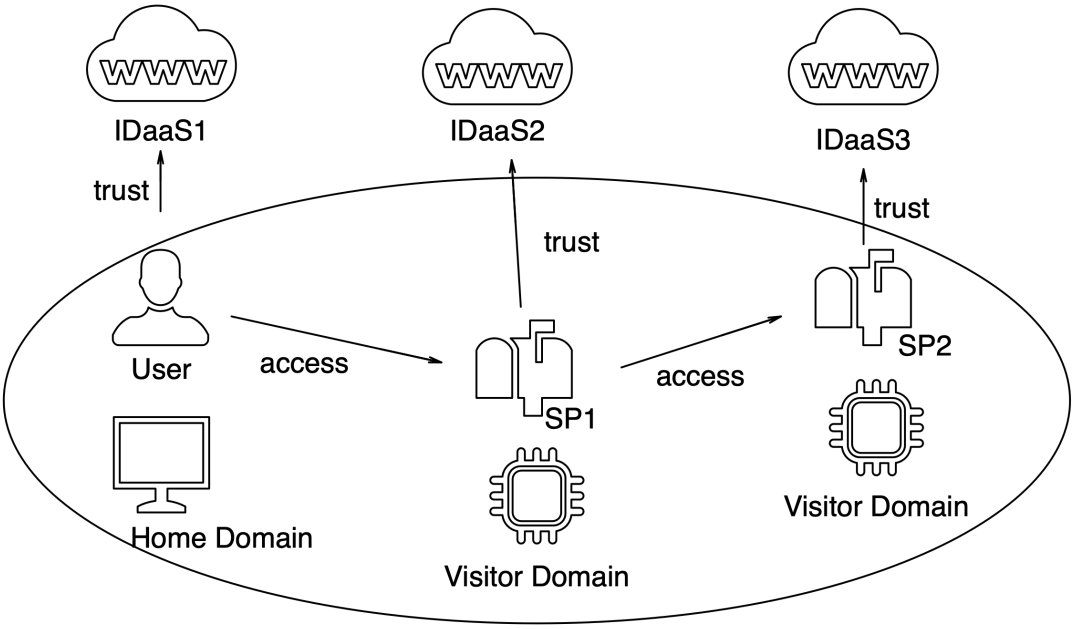


Figure 4: Trust Model - Identity federation

Figure4 shows the trust model they proposed. Every domain owns an Identity Provider(IdP). The user registers at the trusted home Idp. The IdPs of other domains will cooperate with the home IdP. The service provider will trust the Identity Providers of its domain and use them to handle the sensitive operation. For example, this model includes three security domains: Telekom, Salesforce, and Amazon, and each domain have an IDaaS. A customer registers his home IDaaS for authentication. The home IDaaS would then be federated with other domains, which are called visitor IDaaS. In different domains, SPs trust according to IDaaS as an authoritative resource to handle the authentication requests. To be more specific, SP1 trusts IDaaS1, SP2 trusts IDaaS2. This trust model reduces the complexity for a user and an SP to establish trust with each partner individually(2). If and only if the access purposes of the service can satisfy the user proposed before, the IDaaS in the domain can decrypt.

To adapt the security infrastructure for cloud services, their approach considers the separation of duties between a security architect, an IDaaS, and application developers(2). The application developers can only use the user attributes which are available from the environment.

To dive deeper into the trust adaption concept, they use Topology to Model the infrastructure including Node Types, Capabilities Types, and Relationship Types.

2.2.3 Design Principles of Purpose-Based Encryption

For collection of personal data, the law regulates that it should be under consent and clarify the purpose. Also, this collection of personal data has a time limit and could not be disclosed.

To describe the disclosure policy, this paper proposed an access control model - Purpose-based Encryption, including three main factors: time, purpose, and domain. They use the Predicate Encryption with public index to encrypt the data: each ciphertext is associated with a public index

that describes the disclosure policy(2). While this model can allow users to encrypt and distribute their data in any security domains, SPs without satisfying the disclosure policy cannot obtain the key capabilities to decrypt the data.

To be more specific about the encryption scheme, this encryption mainly includes 3 parts:

- Encryption: group and encrypt the Personal Identifiable information that shares a disclosure policy.
- Generalisation: Before encrypting a value, generalization avoids it from being revealed exactly. Multiple authorities are implemented to process secret keys along with the attributes independently. For the key generation, this design uses the UID concatenating with the SESSIONID as input for tying all key attributes together(2). All key attributes are used because they can prevent collusion attacks between users. With all key attributes, this design can prevent users from trying to combine their different keys to gain more capabilities for decryption.
- Minimisation: It prevents more user data from being disclosed than is necessary for the intended purpose. For example, the process of purpose development shouldn't include any personal information.

2.2.4 Life Circle and Request flow

This architecture design mainly contains two request flows.

Single authority

- Encryption: User attributes are encrypted by the home Identity Provider and stored in ciphertext in a Policy Information Point.
- Authentication and authorization: Use "User-Managed Profile" standard and send requests asynchronously.
- Purpose authorization request: Activate the purpose for accessing the cloud service.
- Identity propagation: If the access purpose of the backend service complies with the disclosure policy, it is possible to decrypt the ciphertext.

Multi-authority

- Encryption: Use Multi-authority Attribute-Based Encryption which is mentioned above.
- Authentication: Authenticates the user and generates a Time Access Token using the key generation algorithm.
- Authorisation: The user provides the SAML to access a service provider after authenticating

2.3 Virtual Identity frameworks based on the Identity-Based Encryption (IBE) and Pseudonym-Based Encryption (PBE)

Gomaa *et al.*(18) proposes a new identity concept called Virtual Identity, which can assure security, privacy, reliability, and cost-effectiveness towards a feasible cloud-based IDaaS solution.

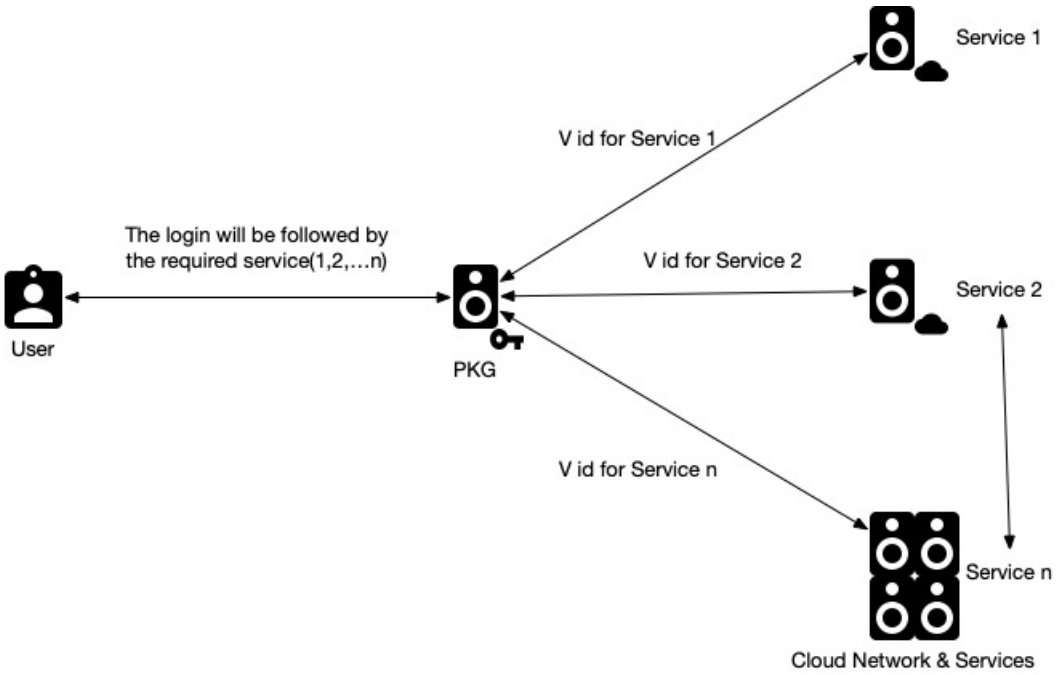


Figure 5: main framework

This new model can make each user have not only one main identity but also different virtual ones. Virtual identities would map into each service access as shown in Fig5 framework.

2.3.1 Proposed Virtual Identity approaches

This framework can create user virtual identity-based IBE and PBE to build connections with cloud service providers. This design contains three main entities: User, Service Provider, and the Private Key Generator.

The basis for creating Virtual Identity includes two secure mechanisms, the first one is based on IBE and the second is based on a different method, PBE. For achieving a high degree of security and efficiency, the IBE and PBE are designed and implemented based on Elliptic Curve Cryptography (ECC)(18). They are different encryption techniques, and both are not anonymous to the Trust Authority. Thus, this design can ensure the identity can develop into Virtual Identity and secure reliability and privacy. Moreover, the two different method shares the same security requirements to secure the system's privacy.

For implementing and designing proposed protocols and further evaluation, 'MIRACL library' is used in the IBE and PBE. The function can calculate the number of points in a finite field which should be a prime number](18).

In this paper, the authors also compare the two approaches – IBE and PBE. While they are both used as solutions for anonymous communications, there are some differences in their processing time and scalability. The processing times for all messages are around 0.05 Sec both IBE and PBE. For the scalability issue, they evaluate the time needed to create the Virtual Identity for the different numbers of users as a simulation scheme for a large number of cloud access users, and the PBE provides a short time for V ID creation because of the low number of messages used to create it(18).

3 Discussion

In this section, a comparison between the privacy enhancement methods suggested by Rupa's, Fuhrmann's, and Gomaa's team would be made. We will then discuss the method models based on the algorithm used and the execution performance.

3.1 Algorithm Used

SAML is utilized for authentication and authorization identity exchange in all Rupa's, Fuhrmann's, and Gomaa's method, because SAML can offer data communication among various types of devices, making it ideal for IDaaS, which demands a lot of communication between users, IdPs, and SPs.

In addition, Rupa's method used PECC as a key encryption scheme. When compared to the other two approaches discussed in the study, PECC may help conceal data (IDs, tickets, and tokens) transmitted via the cloud, whereas the other two methods only hide user identities but not the data itself (1). Therefore, we can say that Rupa's method can better prevent ID theft attacks, reply attacks, and forgery attacks because of the hidden user identities technique.

Fuhrmann's method proposed a Purpose-Based Encryption as their encryption scheme. To the best of knowledge, this work is the first approach to combine Purpose-based Access Control and Attribute-based Encryption to protect the confidentiality of disseminated data with multi-authorities support(2). They prove their concept in real development and solved the following issues successfully. With their proposed Purpose-based Encryption, propagation between intermediaries and further identity theft can be prevented. Besides, Insider attack on an IdP: PII is encrypted on the IdP. Therefore, an administrator of one IdP does not have enough tokens to decrypt it and malicious hosts can be prevented(2).

Gomaa's method did not focus on proposing a new-design encryption mechanism. They come up with a virtual identity technology to reach the goal of preventing the reverse of access chain in the virtualized environment through hiding the main user identity. Their method can also be used for big data secure access mechanisms combined with cloud computing to solve scaling problems.

A table listing the algorithms used in these three privacy enhancement methods is stated inTable 2.

Table 2: Algorithm Used in Different Methods

	Identity Exchange	**Data Encryption**
Rupa's	SAML	PECC
Fuhrmann's	SAML	Purpose-Based Encryption
Gomaa's	SAML	/

3.2 Performance

Aside from the different data privacy enhancing approaches used in the IDaaS model, the actual performance of the cloud environment's communication is also a key concern. Even though a technique offers a high level of security, if it takes a lengthy time to finish the entire authentication process, it isn't always the best option. A performance analysis was carried out in these three studies by assessing their response time, latency, and other elasticity measurements.

However, there is a slightly different in the computer configuration that the three studies used in their analysis, therefore, there may be some comparison errors due to the incompatibility of the configuration applied. Rupa's method measured the performance using the Ubuntu 16.04 LTS and Linux kernel version 3.13.0 environment, while Fuhrmann's method tested on CPU i7-8650U, 1.90 GHz, 16 GB RAM, and Gomaa's method used memory 4GB in Linux Ubuntu 12.10.

- **Latency**

  The latency time in Rupa's method stands for the delay between the communication of userA and SP, this does not include the data processing time.

  Rupa's and Gomaa's methods receive an average of 2 seconds of latency which is much lower than Fuhrmann's method, which is around 12 seconds. The relatively high latency in

Fuhrmann's method relies on the time it spends on verifying the timestamp, purpose, and domain.

- **Response Time**

In Rupa's method, we analyze the performance of response time according to different data sizes transmitted over the cloud. The time it takes for user A to send a request and to receive a service from the SP is shown inTable 3. It's not hard to tell that although Rupa's method only transfers a small amount of data, for example, 50kb, it also takes nearly 2 seconds for the model to respond. Furthermore, the response time grows exponentially as data size increases. Comparing to Fuhrmann's method that only requires 0.19 seconds for transferring 1 MB of data, Rupa's method does have a lower efficiency. This could be due to the larger scale of communication; recall from Rupa's model that both the identity server and the authentication server require an additional step to verify the user's identity.

Table 3: Response Time of Rupa's method

Data Size (kb)	**Response Time(sec)**
10	0.6213
20	0.4842
30	0.7650
40	0.8243
50	1.83

In Fuhrmann's method, they take security level into account. The security level is a standard recommended by the National Institute for Standards and Technology (NIST) on the sizes for secure settings of parings(19). They tested the performance with different data sizes and with different security levels. The result shows that the security level would barely affect the performance. To sum up, for 80 security bits, it only took 50 ms and 37 ms to encrypt and decrypt 1MB, respectively, which is pretty fast. The token generation time is shown in Fig6. However, the performance took a tumble when the data size was above this level.

Table 4: Encryption and Decryption time of Rupa's method

Data Size	**Encryption Time(milliseconds)**	**Decryption Time(milliseconds)**
1 byte	49	33
1 MB	50	37
10 MB	71	62
100 MB	309	328

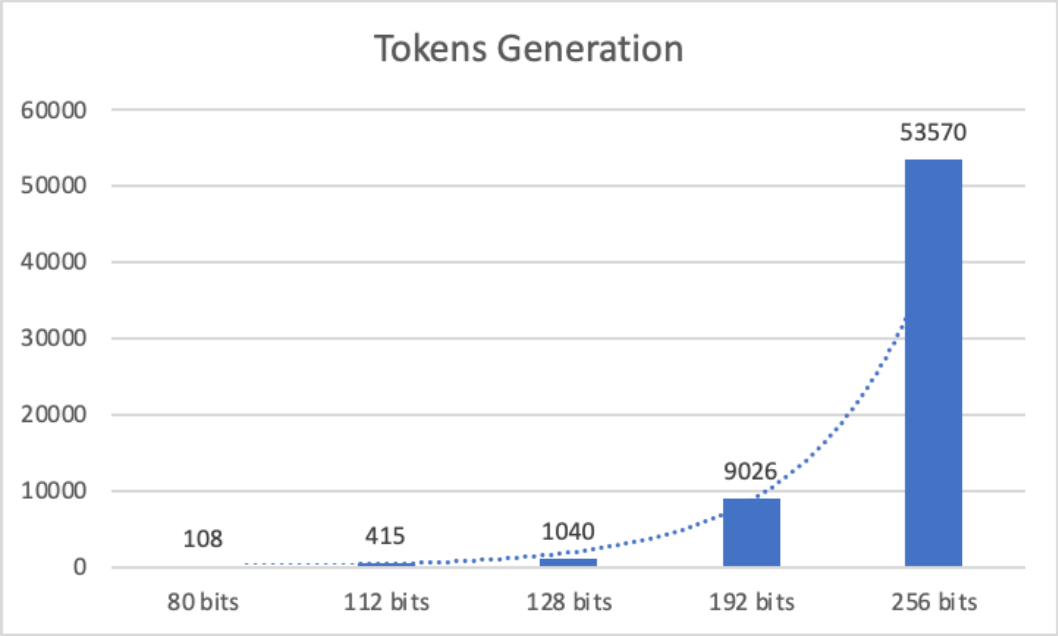


Figure 6: Token generation time

In Gomaa's method, the response time is not included into their evaluation.

- **Authentication load** In Gomaa's method, they compared the authentication load, defined as the number of simultaneous service sessions that a user has(20). The comparison indicated that the proposed Virtual Identification approaches are suitable for cloud-based SaaS applications(18).
- **Security** In Gomaa's method, they analyze the security comparison between their proposed approaches and the same related works. The proposed approaches passed three out of four AVISPA back-ends (OFMC (On-the-fly ModelChecker), CLAtSe (CL-based Attack Searcher), SATMC (SAT-based Model-checker), and TA4SP (Tree-Automatabased Protocol-Analyser))(18). Thus, this method is more secure compared with other competitive schemes.

3.3 Limitations

The three papers all offered a solution to the cloud-based IDaaS privacy problem. However, the authors of these studies also point out some of their IDaaS models' limitations.

Rupa's method (1)pointed out that the model's performance is insufficient due to the multiple identity verification steps(See steps 2,8,5,11,14 in Figure 3). Furthermore, Rupa's method may place a heavy communication load on the cloud environment because each of their steps generates a timestamp. Once the timestamp expires, the whole communication needs to start over again.

Fuhrmann's method(2) cannot prevent the collision attacks between the home and visitor IdP, despite the authors assuming adversaries cannot control both IdPs at the same time.

Gomaa's method(18) would create virtual identity, so the authentication load can be an issue. How to improve the correlation with the keystone security server and further better integrate with cloud scalability would also be challenging for this method.

4 Conclusion

In this literature review, we investigated the architecture of IDaaS, and looked into how it develops from cloud-based services. It is noted that the difference between IAM and IDaaS is that IDaaS deployed its identity services on the cloud. IDaaS, however, faces many privacy challenges. In the literature survey, we analyzed the three privacy enhancement methods for IDaaS. Some findings are shown below:

- **Identity protection method**
  Both Rupa's and Fuhrmann's methods used an encryption technique to encrypt information transferred between users, IdPs, and SPs, with the goal of concealing the information itself. Gomma's method, on the other hand, focuses on concealing the user's identity rather than the information transmitted by creating a virtual identity.
  Furthermore, all three studies use SAML as their identity protocol, demonstrating that SAML is the best option for IDaaS when compared to other identity protocols such as OAuth.
- **Latency**
  When comparing the three approaches, Fuhrmann's study had the highest latency due to the time spent confirming the timestamp, purpose, and domain, while Rupa and Gomaa's methods had the same latency in communicating identity information.
- **Response time**
  Fuhrmann's method responds in a short time for the total time it takes for the user to receive the service by efficiently processing the encrypting, decrypting, and token generation procedures despite the large data size. However, due to the verification step in the protocol model, the response time of Ropa's method grows exponentially with data size increase.

The limitations of each method are also analysed afterwards. Further upgrading the models for better solutions will be a realistic and sustainable purpose.

5 Work distribution

ChiaYu Lin is responsible for part of section 1.2, section 1.3, section 1.4, part of section 2, and part of section 4

Jianyang Gu is responsible for section 1.1, part of section 2, part of section 3, and part of section 4

Wenjun Liang is responsible for part of section 2 and part of section 3

References

[1] C. Rupa, R. Patan, F. Al-Turjman and L. Mostarda, "Enhancing the Access Privacy of IDaaS System Using SAML Protocol in Fog Computing," in IEEE Access, vol. 8, pp. 168793-168801, 2020, doi: 10.1109/ACCESS.2020.3022957.

[2] Vo, T. H., Fuhrmann, W., Fischer-Hellmann, K. P., & Furnell, S. (2019). Identity-as-a-service: An adaptive security infrastructure and privacy-preserving user identity for the cloud environment. Future Internet, 11(5), 116.

[3] N. Naik and P. Jenkins, "An Analysis of Open Standard Identity Protocols in Cloud Computing Security Paradigm," 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2016, pp. 428-431, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.85.

[4] Oracle - What is SAML, https://www.oracle.com/in/security/cloud-security/what-is-saml/

[5] I. Indu, P.M. Rubesh Anand, Vidhyacharan Bhaskar, Identity and access management in cloud environment: Mechanisms and challenges, Engineering Science and Technology, an International Journal, Volume 21, Issue 4, 2018, Pages 574-588, ISSN 2215-0986

[6] Mohammed I A. Intelligent authentication for identity and access management: a review paper[J]. International Journal of Managment, IT and Engineering (IJMIE), 2013, 3(1): 696-705.

[7] https://solutionsreview.com/identity-management/cloud-identity-management-differ/

[8] Cox P. How to manage identity in the public cloud[J]. InformationWeek reports, 2012.

[9] https://www.ibm.com/cloud/learn/iaas-paas-saas

[10] Fisher, W., Brown, C., Russell, M., Umarji, S., Scarfone, K. (2021). Identity as a Service for Public Safety Organizations (No. NIST Internal or Interagency Report (NISTIR) 8335 (Draft)). National Institute of Standards and Technology.

[11] Understanding IDaaS: The benefits and risks of Identity as a Service, Tech Target, https://www.techtarget.com/searchsecurity/feature/Understanding-IDaaS-The-benefits-and-risks-of-Identity-as-a-Service.

[12] Samonas, Spyridon, and David Coss. "The CIA strikes back: Redefining confidentiality, integrity and availability in security." Journal of Information System Security 10.3 (2014).

[13] Josang, A., AlZomai, M., Suriadi, S. (2007). Usability and privacy in identity management architectures. In ACSW Frontiers 2007: Proceedings of 5th Australasian Symposium on Grid Computing and e-Research, 5th Australasian Information Security Workshop (Privacy Enhancing Technologies), and Australasian Workshop on Health Knowledge Management and Discovery (pp. 143-152). Australian Computer Society.

[14] Kadam, Yashpal. "Security issues in cloud computing a transparent view." International Journal of Computer Science Emerging Technology 2.5 (2011): 316-322.

[15] Nuñez, D., Agudo, I. BlindIdM: A privacy-preserving approach for identity management as a service. Int. J. Inf. Secur. 13, 199–215 (2014). https://doi.org/10.1007/s10207-014-0230-4

[16] Nikhila, V. & Rupa, Ch. (2019). Intensifying Multimedia Information Security Using Comprehensive Cipher. 1-4. 10.1109/i-PACT44901.2019.8960002.

[17] HowtoManage Identity in the PublicCloud, Information Week Report, https://dsimg.ubm-us.net/envelope/280082/485573/strategy-how-to-manage-identity-in-the-public-cloud_1165433.pdf

[18] Gomaa I, Abd-Elrahman E, Saad E, et al. Virtual identity performance evaluations of anonymous authentication in IDaaS framework[J]. IEEE Access, 2019, 7: 34541-34554.

[19] Barker, E. Recommendation for Key Management—Part 1: General. In NIST Spec. Publ. 800-57; National Institute of Standards and Technology: Gaithersburg, MA, USA, 2016; pp. 1–142.

[20] M. Barisch, “Modelling the impact of virtual identities on communication infrastructures,” in Proc. 5th ACM Workshop Digit. Identity Manage., New York, NY, USA, Nov. 2009, pp. 45–52. doi: 10.1145/ 1655028.1655040.

Serverless Computing and Function as a Service: Security Perspective

Behnam Bozorgi
Department of Computer Science
University of Amsterdam
14125056
behnam.bozorgi@student.uva.nl

Kimiya Ataiyan
Department of Computer Science
University of Amsterdam
13582453
kimiya.ataiyan@student.uva.nl

Abhilash Balaji
Department of Computer Science
University of Amsterdam
14476347
abhilash.balaji.balaji@student.uva.nl

Abstract

This literature study highlights the matter of security related to serverless computing and FaaS. The measures of security have gotten more complex with serverless computing and FaaS, due to the separation between provider and user, and several isolations and independence that are used to make applications more scalable and efficient. First, we define important components to understand the content of the literature, then we highlight existing challenges and possible solutions to these challenges, and lastly finish with security measures that are taken in the cloud provider industry. Security is essential to maintaining reliable applications within an organization.

1 Introduction

Cloud computing has become an inevitable tool to use in industry and also an interesting research topic these years. Its evolution brought many innovations to research and industry including serverless computing. According to a study in 2020, the market worth of serverless computing will reach more than 21 billion dollars by 2025, at a Compound Annual Growth Rate (CAGR) of 22.7% during the forecast period [24]. Cloud computing is a service that can provide storage, applications, and processing power over the internet. Its technology part consists of IaaS[1], PaaS[2], SaaS[3], and Serverless.

Serverless computing is based on dividing the workload into small pieces to achieve auto-scaling and fast deployment. This way the "pay-as-you-go" model works since you are only charged for the pieces and resources that you are actually using and running. Part of the serverless model is FaaS (function-as-a-service), in which cloud users are able to develop, run and manage their applications, without managing the infrastructure and architecture of it. FaaS is based on the execution of small and lightweight functions that each performs a single logical task. In this scenario, developers are not in control of where and how this data is flowing, which creates a risk to security [8]. So, despite FaaS's benefits such as auto-scaling, cost reduction, and availability, serverless computing also brings

[1] Infrastructure as a Service
[2] Platform as a Service
[3] Software as a Service

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

limitations and problems, security being part of them [26]. Conceptually thinking about security, serverless platforms should be capable of securing functions from attacks.

The main contribution of this work to security in serverless computing are as follows:

1. Bring discipline and categorization to challenges of security in serverless computing.
2. Go over recent reviews of academic and also industrial solutions for challenges.
3. Introduce the most important research on challenges and current solutions to them.
4. Present our perspective on this topic and future research potential.

2 Definitions

In this topic definitions of topics are presented to make the enthusiasts and researchers have a basic understanding of general concepts in cloud computing.

2.1 Serverless computing

Serverless computing is a relatively new concept and technology which started its trend from 2016 based on google trends over the internet. But there were few interests around 2004 which were taken seriously. Figure 1 shows the amount of interest in serverless computing over the past 6 years on google trends.

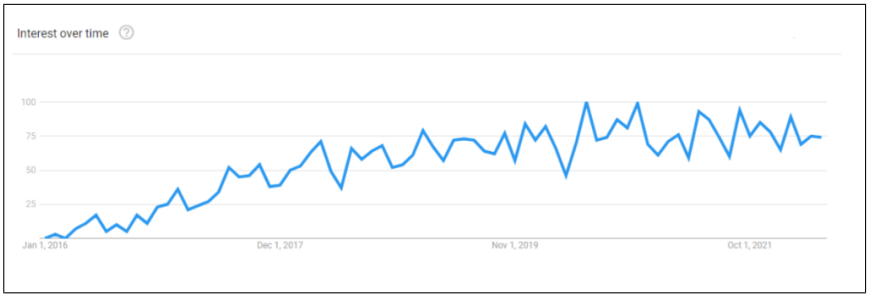


Figure 1: Trends of serverless computing

For defining serverless computing we need to look at how much control a developer has over cloud infrastructure. For that matter as it is shown in Figure 2, IaaS gives the most control application which is coded and configuration of infrastructure in the cloud. In this model the developer can configure how to deploy and run the application and can customize the process. On the other hand, In SaaS developer is not even aware of infrastructure therefore configuration and customization can not be done in infrastructure and developer only allow to host code here which is strictly coupled [6].

In serverless computing, the cloud provider dynamically allocates and provisions servers. The code is executed in almost-stateless containers that are event-triggered, and ephemeral (may last for one invocation). Serverless covers a wide range of technologies that can be grouped into two categories: **Backend-as-a-Service (BaaS)** and **Functions-as-a-Service (FaaS)** [20].

2.2 Backend-as-a-Service

Backend-as-a- Service allows for the replacement of server-side components with off-the-shelf services. BaaS allows developers to outsource all aspects of an application's backend, allowing them to write and maintain all application logic in the frontend. Remote authentication systems, database management, cloud storage, and hosting are some examples. Google Firebase, a fully managed database that can be used directly from an application, is an example of BaaS. Firebase (the BaaS services) manages data components on our behalf in this case. [20].

2.3 Function-as-a-Service

Function-as-a- A service is an environment in which software can run. Serverless applications are event-driven cloud-based systems in which application development is entirely dependent on a combi-

nation of third-party services, client-side logic, and cloud-hosted remote procedure calls[**ref 23**].FaaS enables developers to deploy code that, when activated, executes in a separate environment. Each function typically describes a subset of an entire application. Function execution times are typically limited (e.g. 15 minutes for AWS Lambda). Functions are not constantly active. Instead, the FaaS platforms watch for events that instantiate the functions. As a result, functions must be triggered by events such as client requests, events generated by any external systems, data streams, or others.

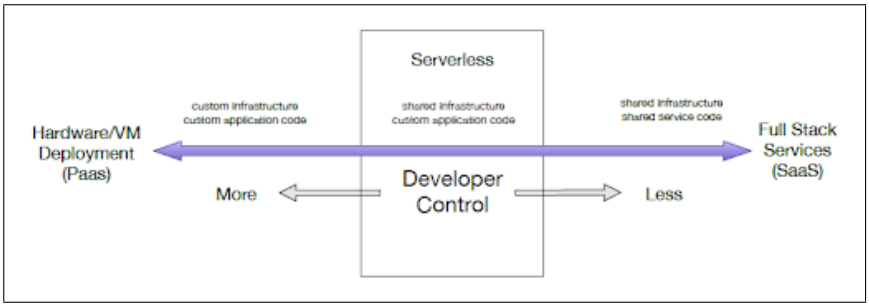


Figure 2: Range of developer control over cloud infrastructure
[7]

Serverless computing is somewhere in the middle. In serverless computing developers has control on code but they need to write code in a certain way called "stateless functions". It is important that enthusiasts do not fall for the misconception of naming serverless computing because servers indeed are needed but developers need to understand that they are responsible to manage them. Some parameters like the capacity of the server are configured by the cloud platform depending on the workload on the server or auto scaling and fault tolerance of deployment are guaranteed by the platform. So developers only need to code somehow independent of where it is going to run [6].

Lastly, we want to highlight the differences between serverless computing and server computing, before diving into the security of serverless computing and FaaS as shown in 3.Before the emergence of serverless architecture, developers faced manu challenges when setting up a cloud environment for applications. The traditional IaaS infrastructure showed challenges in scalability, and thus the demand for an architecture that is able to scale up or down depending on number of request by a module rather than application. In addition, orignally organizations relied on their large-scale applications to be hosted in each region of the world, which becomes very expensive for an organization. Serverless computing on the other hand solves this issue by loading up the code closest to the origin of the request. Lastly, the traditional approach showcased issues with deploying updates to the server, because it was time consuming and required specialized developers to update existing code. With serverless architecture, developers no longer make backend changes but rather only upload the new update in patches [21].

3 Challenges and possible solutions

Security is one of the main concerns in serverless computing research in recent years and there are some researches that focus on different topics in security. In this section, we tried to categorize challenges in the security of serverless computing and present important works in each category. As illustraited in Table 3, challenges in security of serverless computing can be categorized in four different categories:

Table 1: Summary of security challenges categories and solutions

Challenge	**Solution(s)**	**Important works**
Data Protection	API Protection, Code Protection (Intel SGX), API Authentication (e-Lambda)	[17],[26],[16],[11], [22],[21],[8],[9]
Resource Isolation	Virtualization, Containers,Language-specific Isolation	[9],[1],[13]
Security Monitoring	Dynamic tracing, Proxy network requests using agents	[13],[8]
Security Management	IAM, PKI	[17], [23]

Property	Types of Computing	
	Serverless Computing	***Server Computing***
Physical Machines	Code runs in containers that are not limited to a specific machine.	Code is run in a specific instance only
Deployment Time	Quicker to deploy	Takes more time to deploy
Scalability	The architecture scales itself depending on the demand automatically	Scalability needs to be set up manually by the developer
Price	Charged only for the time code is running	Charged for the time the machine is active
Granularity of Code	Very high	Less
Cold Starts	Yes	No
Latency	Reduces latency as the code is run closer to the user	Depends on the location of the instance running the code.
Fault Tolerance	Very high	Low
Testing/ Debugging	Complex	Comparatively easier
Type of applications	Suitable for small and medium-scale applications	Suitable for large scale projects

Figure 3: Comparison of serverless and server computing
[21]

3.1 Data Protection

In the FaaS model of serverless computing, users create and load code in the cloud environment while only paying for CPU time, storage, and network operations required to run and execute the code Since this shifts the responsibility of server/container security to the provider, an organization only needs to be responsible for the security related to their code and its functionality [17].

In FaaS functions are independent of one another and their run-time is relatively fast since they usually are small functions of the application - although, certain providers such as AWS set an upper bound for function execution time [26]. Another characteristic of FaaS is the transparency which provides the capability of separating the user from code deployment and management, but rather only requires the user to upload their function. The functions written by the cloud user are stateless, which ensures robustness in case of crashing. As mentioned above, a huge benefit of serverless computing and FaaS to organizations is the “pay-as-you-go” model, in which the user is only charged for the uploaded and executed functions [26].

While it may seem that organizations moving to FaaS require less attention to the security of the organization, this only holds true considering the security of the host OS and servers. Cloud providers provide security functionalities such as cloud identification and authorization management (IAM), policy and role management, and access control management [17]. On the other hand, the organization is fully responsible for the security of the application and the code that holds it together, which includes validation and communication of data and library calls to third-party code or services [17]. A large number of functions, using API calls can lead to a loosely coupled setup and thus open a door for inappropriate access by APIs. In addition, the issue of validation of input becomes more difficult as the validation may be spread across multiple functions that each needs its own input validation. It commonly happens that developers make the mistake of their application being exposed to APIs without correct authorization controls, allowing an attacker to bypass and access the codebase and application. Since FaaS architecture is designed after a stateless web design, OWASP (Open Web Application security project) web vulnerabilities are also possible issues in FaaS [17]. Thus security issues with FaaS include Injection, sensitive data exposure, broken authentication, XML external entities, broken access control, security misconfiguration, cross-site scripting (XSS), insecure deserialization, and using components with known vulnerabilities, and insufficient logging and monitoring [17].

Many of the public cloud providers provide APIs to programmatically handle working with their products that provide FaaS, and with cyber attacks doubling 5 years [16] open cloud vendors must make sure to harden their public facing APIs as a form of threat prevention, so that no bad actors can enter the platform through over privilege or XSS.Scientists have found multiple issues with the current design , architecture and documentation of client facing APIs of leading cloud providers leading to services being misconfigured due to a lack of clarity on the APIs or actual steps required for hardened cloud policies being omitted altogether [11].

Organizations are offered a high degree of flexibility with little to no effort in moving code between FaaS providers, due to the shared practices of supported functionality subsets among providers. This high degree of flexibility comes with the price of the user needing to maintain the security of the entire communication flow of functions as well as the state datastore. All Linux container instances share the OS, which causes all containers to be vulnerable to breaches and tampering as soon as one of the containers fails [26]. An example of this vulnerability is the published shocker[8] attack which several containers instances can access one another as well as files on the OS caused by a bug in system calls to file handling in just one of the containers. Another example of vulnerability due to shared OS is in the multitenancy phenomenon. Here security can be at threat when for example two instances running on the same application can access one another's data because the customers is using the same resource for each of their applications. This also means that one user's error may cause an error for other users due to lack of isolation [21].

Going beyond the challenges of data protection related to FaaS, a major and general challenge in serverless computing security in data protection is securing the API gateway. In a typical application like a web or mobile application, there are APIs to expose a service to the outside world. There are best practices that include tokens in APIs to make it more secure and the gateway is responsible to obtain this token. That is why these API gateways are attractive targets for attackers [22]. In [22] they proposed a solution for gateway security called Se-Lambda. This solution includes two main actions which are Core Modules Isolation and Function Validation. Figure 4 shows the architecture of Se-Lambda.

Each API in Se-Lambda has core modules like authentication. When a request reaches the API, first it is parsed and checked if there are any security concerns for malicious attacks. After the request is scanned and approved in terms of security, the user authentication module is called, and the token is controlled to prevent any unauthorized access. At the final stage, The API gateway calls the appropriate service runtime to run function modules and returns the results to the user. Se-Lambda also does not accept other software on the API gateway not only because of security reasons but also because putting many modules in the API gateway results in large TCB[4] which increases the overhead. Se-Lambda only places privacy-sensitive modules in the Intel SGX enclave to isolate them from being exposed to threats [9].

By making use of hardware attestation, Se-Lambda uses SGX's remote attestation to perform integrity checking for function modules. Se-Lambda does integrity validation on modules to make sure those

[4]Trusted Computing Base

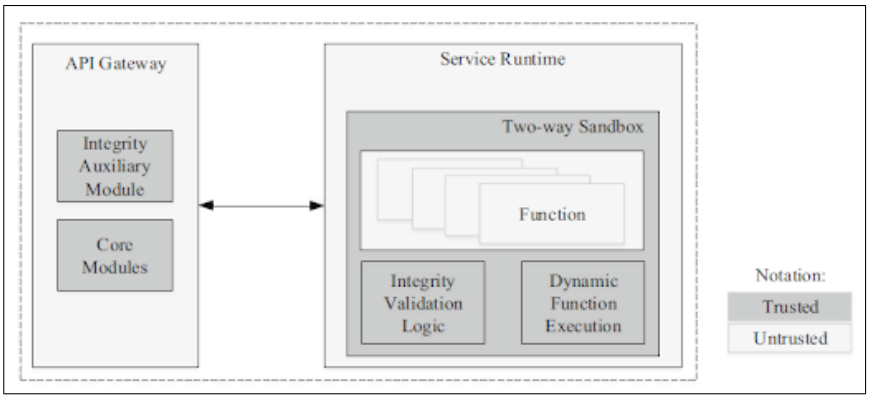


Figure 4: Architecture of Se-Lambda

modules are the ones that were supposed to be running. Se-Lambda also provides a service runtime that makes use of a two-way sandbox to make function modules more protected. The sandbox provides two-way protection: the SGX enclave protects the confidentiality of users' data, and the WebAssembly sandboxed environment protects the security of the cloud provider's host runtime.

3.2 Resource Isolation

Another challenge is having a trade-off between security and performance which happens in resource isolation [9, 1] . This challenge is mostly concerned with the isolation option that an application uses. Mainly there are three different ways to bring isolation to workloads on Linux: Virtualization, Containers and Language-specific Isolation [1] . Figure 5 shows the different security approaches between containers and virtualization. Linux containers rely on the kernel's sandboxing features directly, whereas KVMstyle[5] virtualization relies on the VMM's[6] security, maybe with augmented sandboxing.

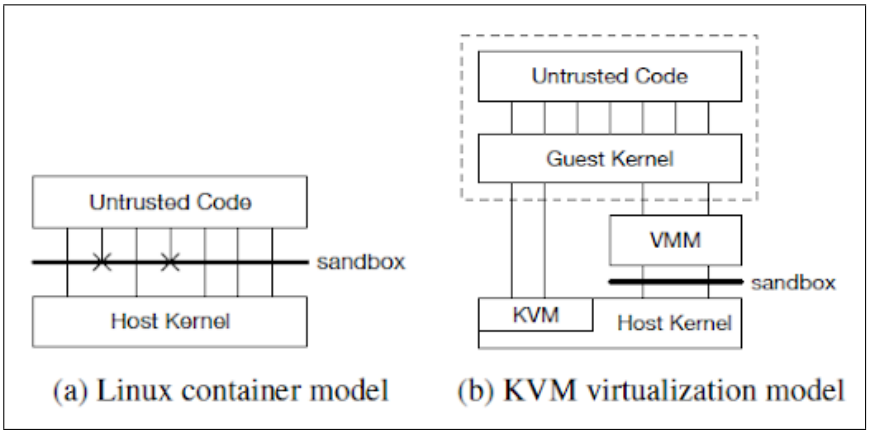


Figure 5: Security approaches between Linux containers and virtualization [1]

The first category which is the oldest is virtualization. In virtualization, functions run in their isolated VMs under a VMM. Untrusted code is often given full access to a guest kernel in virtualization, allowing it to exploit all kernel features while expressly designating the guest kernel as untrusted. Virtualization brings isolation on the hardware level as well, but it comes with two main challenges: density and overhead. But with these challenges, the great benefit of virtualization in a perspective of isolation is that it shifts the security-critical interface from the OS border to a hardware-supported and

[5]Kernel-based Virtual Machine
[6]Virtual Machine Monitor

comparably simpler software boundary. Virtualization eliminates the need to choose between kernel functionality and security: the guest kernel can provide its entire feature set while the threat model remains unchanged. VMMs are substantially smaller than general-purpose OS kernels, exposing only a few well-understood abstractions while preserving program compatibility and avoiding the need to modify applications. The second one is containers which is a relatively newer technology. In containers, the kernel is shared among workloads and the kernel provides possibilities to isolate them. Untrusted code in Linux containers communicates directly with the host kernel, maybe with the kernel surface area limited. It also has direct interactions with the host kernel's other services, such as filesystems and the page cache. Containers provide many features for security isolation like cgroups to provide resource (CPU, memory and etc) limitations, namespaces provide isolation for kernel resources like process IDs or network resources. But, seccomp-bpf brings the most important security isolation. seccomp-bpf allows a process to specify which syscalls it can use and the arguments it can provide to these syscalls. The last method for isolating workloads is using a language virtual machine like JVM[7]. Some of these language VMs intend to run multiple workloads in a single process. This kind of language VMs imposed a tradeoff between functionality and security including resistance to side-channel attacks such as Spectre. Inside the cloud, physical co-residency is at the heart of hardware-level side-channel attacks [13]. Based on this fact, if physical co-residency is prevented, it may conflict with startup time, resource utilization, and communication optimization [13].Other approaches tend to use a process per trust domain and stop the code from escaping the process boundaries. It is worth mentioning that this kind of isolation is not suitable for Lambda because in Lambda there is a need to support arbitrary binaries. In [1] Amazon team presents a new open-source VMM called Firecracker which is designed specifically for serverless workloads but also helpful for containers, functions, and other computational workloads that fit under a fair set of limitations.

3.3 Security Monitoring

Another field of security challenge in serverless computing is security monitoring. Communication between cloud functions can expose access patterns and time information. Unlike serverful applications in which data are transmitted in batch and cached locally, Cloud functions are widely distributed across the cloud. So, in this data transmission more sensitive data may be leaked even in encrypted transmission. In this scenario, transforming serverless applications into multiple small functions makes the security exposure even more. It is true that most threats come from outside of a company but protection from internal threats is not to be taken lightly despite the fact that this kind of protection has more overhead [13]. A typical serverless platform has certain runtimes for functions but also supports customized runtimes to support a variety of customized needs. There is also a possibility of having common packages of specific programming languages, but security threats arise when a third-party package comes to play. That is why a reliable monitoring feature is needed to detect abnormal behavior and always check the security status. One approach to tackle this challenge is to do dynamic tracing. In [8] used an agent to proxy network requests, an agent was inserted in each function's container. These agents produced and send out tags in every network traffic so they can be tracked and also limit function behavior based on a specific configuration. The architecture of Valve is shown in Figure 6.

These agents consist of 3 different profilers. The first profiler is Network Profiler which is designed to look at the request header and check the tags are correct according to the current data flow. Since the containers can contain any kind of programming language, REST-based API calls. The second profiler is API Request Validator. Its sole job is to check the request with application security policies to see if anything is against the policy, it drops the request. In the worst-case scenario, an access request is denied which only causes a single web session workflow to fail. So, this validator makes sure the stability of the system is intact as well. And the third profiler is called File Access Profiler. In this profiler, there is a tracer that keeps track of which files are accessed by any kind of file operations in the code and after function execution is finished, all data on the disk that was modified is deleted. The current attack mechanism is about data flow from one function execution to another and by deleting all modified files, the chain of hopping from one function execution to another is stopped. The garbage collection specifically exists to prevent this cross-invocation to deny attackers from manipulating the system [8].

[7] Java Virtual Machine

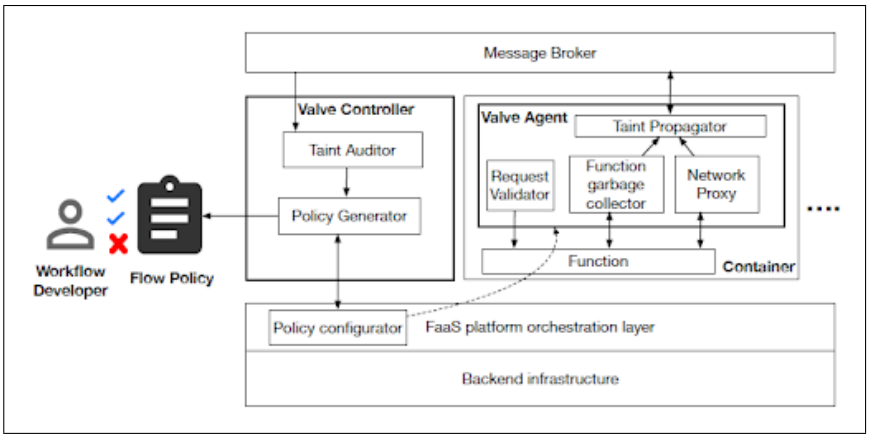


Figure 6: Architecture of Se-Lambda

3.4 Security management

The following security issue mitigations are related to OWASP[8] issues. The first problem of injection in FaaS, when unverified data can be passed between functions, can be mitigated by using a safe API – one that does not use interpreters – as well as having each function validate input and encode output. In cases where a developer uses FaaS for user authentication, the provider's FaaS IAM (Microsoft Azure Active Directory, Amazon's IAM [9] implementation) should be leveraged rather than the developer doing their own authentication, since this would require the establishment of authenticated connections and minimal permission for data flows. The risk of faulty code and giving wrong or too loose access is a security issue itself, thus it is safe to use existing tools from the provider [17]. In order to maintain protected data via encryption and decryption, organizational IAM or PKI[10] functionality can be leveraged for managing keys that are used between function communications. In addition, functions should be assigned to trust groups, where communication between trust groups with different labels each have their own unique keys, so that communication between same-label-trust-groups have common keys [17].

The security issue related to broken access control appears when developers make use of authorization tokens that are overly capable – for example setting permissions on group level. Instead, permissions to data should be ACLed[11] so that functions are not able to iterate through the state information store but rather get passed a hashid to retrieve a specific information [17]. Since FaaS is structured in a way with a large number of independent functions, there is an increase in vulnerability to attacks from surface, also known as cross-site scripting. Each function should be safeguarded against XSS[12], since losing control of dataflow can expose functions that are not meant to be exposed. Preventing unwanted exposure of functions can be mitigated by using trustworthy encoding libraries to encode data and validate input. The constant passing of data between stateless functions requires that objects are serialized during sending and deserialized during receiving. To do this in a secure manner digital signatures can be used on transmissions to identify trusted and untrusted sources during deserialization procedures.

Security issue mitigations that are more so related to the developer keeping versions up-to-date or monitoring the platform's functionality are critical. The developer is responsible for updating patches whenever manufacturers release new versions, and simultaneously double check dependencies to make sure library components are the most recent. In addition, monitoring the functionality of the platform, and acting in unexpected behavior are more so possible if the developer creates code that validation failures and sensitive transactions are logged accordingly [17]

[8] Open Web Application Security Project
[9] Identity Access Management
[10] Public Key Infrastructure
[11] Access Control List
[12] Cross-Site scripting

Another perspective is to see the problem at the entry point. In [23] They control every external requests reach to gateway API for authorization and permission. In this work a new ***WILL.IAM*** framework was introduced. This framework is an access control model which not only obey notions of typical IAM-style role-based access control but also has same design principles of serverless application. WILL.IAM framework consists of three components: API Gateway, Policy evaluation service and the request handler. An external path to the deployed functions is provided by the API gateway, which is embedded into the FaaS platforms. The API gateway in WILL.IAM has been enhanced to send externally generated requests to the policy evaluation service and internally generated invocation requests to the appropriate function instances. The access control policies set for various serverless processes implemented in the cloud are enforced by the policy evaluation service. Each function-instance (i.e. container) has a request handler that transparently modifies will. WILL.IAM is function-agnostic since it includes IAM-specific headers in the invocation request before delivering it to the function. As shown in Figure 7, these three components work together to enforce access control in serverless cloud platforms.

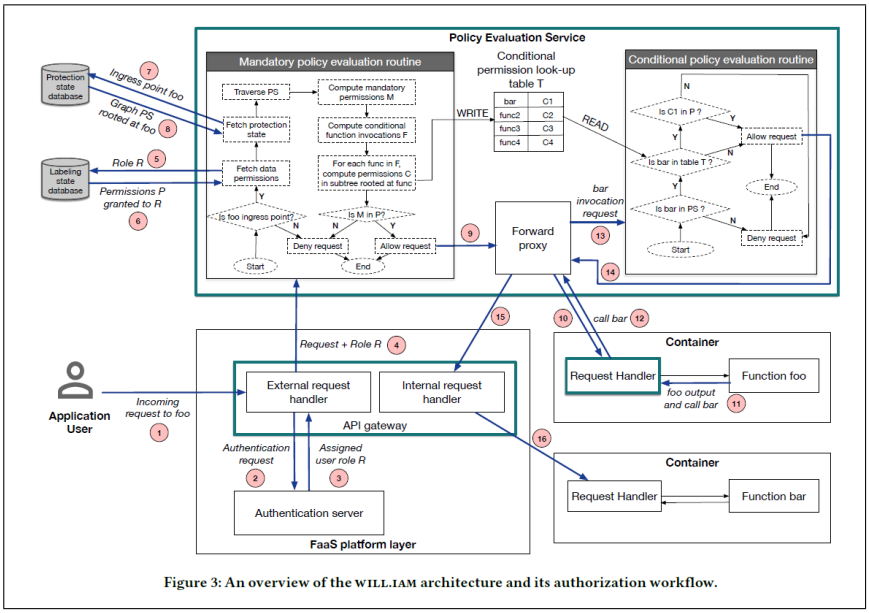


Figure 3: An overview of the WILL.IAM architecture and its authorization workflow.

Figure 7: WILL.IAM architecture and its authorization workflow.

Lastly, the developer should be aware of expected dataflows. Unique certificates can be implemented for known communication paths, which help filter for unexpected communication paths. Certificates help to establish communication between two components that both trust the certificate. Overall, FaaS brings high flexibility and scalability to serverless computing. Yet this means that the more complex and larger a developers platform and application gets, it involves more security measures, thus requiring an organization to be strong in management and process controls [17]. Figure 8 shows several cloud security functionalities provided by AWS, Google and Microsoft, that deal with Access Control Management, Policy and Role Management and Identity and Access Management.

4 Tools - Security measures in industry

One of the challenges mentioned above speaks on resource isolation. In order to achieve isolation of tenants Linux uses approaches based on (for example) Virtual Machines and Containers Virtual Machines use a combination of software-assisted methods and hardware features to achieve virtualization, which moves the security interface from the OS to software and hardware. Generally, a hardware-mechanism is adopted to create an isolated environment with its resources as a sandbox. Containers on the other hand share the same host OS kernel, weakening isolation. Thus, containers include and combine several isolation mechanisms such as namespaces, cgroups, chroot, and seccomp-bpf to create isolation and policies for system calls and process permits. This approach implies that the container relies on system calls from the host machine, once again leading to weak isolation.

Identity and Access Management

- Amazon Web Services
  - AWS IAM (Core AWS identity and access management service)
  - AWS support for Microsoft Active Directory
    - AD Connector – Integrates corporate on prem AD to AWS
    - Simple AD provides commonly used MS AD functions as a managed service
    - Managed Microsoft AD
  - Federation via Identity providers that support OpenID Connect and SAML 2.0
- Google Cloud Platform
  - Google Cloud Identity
  - Federation via Third Party Identity Connectors – these are available for major suppliers
  - Cloud Directory Sync via LDAP
- Microsoft Azure
  - Azure Active Directory (authorization is performed by Azure Graph)
  - Federation via Identity providers that support OpenID Connect and SAML 2.0

Policy and Role Management

- Amazon Web Services
  - AWS IAM (Core AWS identity and access management service)
  - Access policies can be customized and assigned to users, groups, and roles.
- Google Cloud Platform
  - Permissions are applied at the 'Project' level, with users being allowed multiple projects.
  - All permissions apply to all resources within an individual project.
  - Standard roles (RO, RW, Owner) exist and can be customized and applied to users and groups.
- Microsoft Azure
  - Permissions are applied at the Subscription, resource group, and individual resource level. All resources should be in resource groups. Resource groups may be grouped in hierarchies.
  - Common roles are owner, contributor, and reader, but a host of predefined roles exist and custom roles are supported.

Access Control Management

- Amazon Web Services
  - Account keys – god key for account
  - Access keys
  - Temporary security credentials via SAML or OpenID Connect
- Google Cloud Platform
  - Account keys – god key for account
  - Service account keys
  - Storage access token
  - API keys
  - User account tokens
- Microsoft Azure
  - Account keys – god key for account
  - Storage Access keys – root key for storage
  - Connection strings
  - AD auth (user and service)
  - MSI tokens
  - SAS tokens
  - Mutual TLS

Figure 8: Architecture of Se-Lambda

So, several cloud providers such as Google and Amazon have introduced different mechanisms to mitigate existing security issues, including resource isolation.In this section, we go through the tools in serverless computing and security in serverless computing [24]

4.1 gVisor

gVisor is a serverless function introduced by Google to completely isolate the function from the OS by providing a kernel that intercepts and processes all system calls from the function – basically a sandbox for containers [25].Sentry (the kernel) handles memory management and thread scheduling via function calls so that the majority of function calls get handled within the Sentry. Only in a few cases such as I/O operations, the request is forwarded to host Linux services [26].Thus for the above-mentioned challenge of containers sharing a kernel, gVisor provides a separate Sentry for each serverless function, which in return provides better isolation [15].In addition, gVisor achieves the same isolation level as VMs due to its hardware virtualization approach. A downside to gVisor is its lack of support for I/O operations, especially in cases where serverless functions execute such frequently. In Figure 9, the multiple components of gVisor are shown. Each container in the sandbox has its own Gofer, which provides file systems access to containers. The Sentry (kernel) runs the containers and acts a an additional intercept by responding to system calls from the application [25].

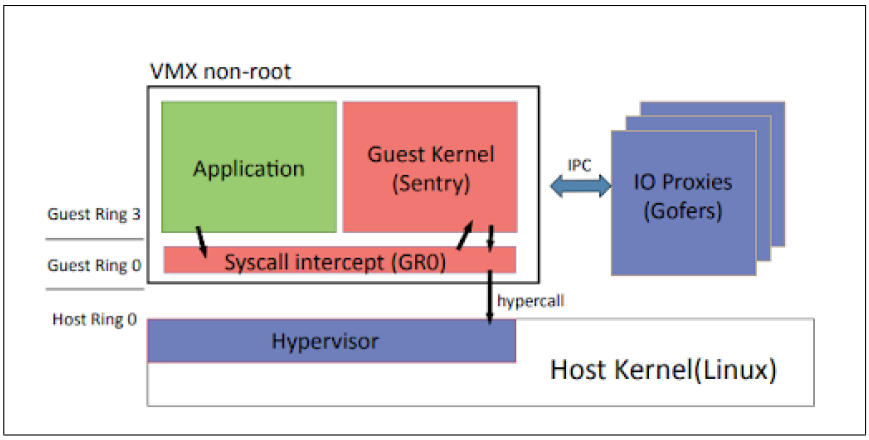


Figure 9: The architecture of Google gVisor
[15]

4.2 FireCracker

Amazon approaches the isolation of containers via the FireCracker which uses two levels of isolation boundaries using hardware virtualization and a jailer box which is based on seccomp and namespace. FireCracker launches light-weight virtual machines called microVM, which hold a container. The microVM has an advantage to the regular container-in-VM approach since it removes all unnecessary functionalities such as unused drivers and thus is more speed and resource efficient [10, 15].Figure 10 shows the architecture of AWS FireCracker, in which the kernel-based VM (KVM) creates and manages microVMs. This architecture showcases that each container group can be held within a virtual machine barrier, enabling workloads to be run on same machine while maintaining security [10].

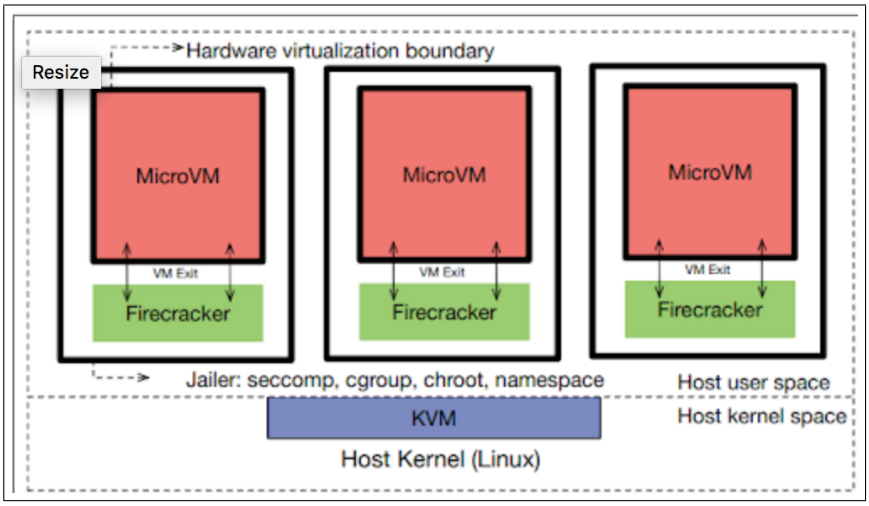


Figure 10: The architecture of AWS FireCracker
[15]

4.3 Intel SGX[13] enclaves

One of the tools which are used in serverless computing security is Intel Software Guard Extensions. Intel Software Guard Extensions (Intel SGX) is an Intel technology that allows application developers to safeguard certain code and data from being disclosed or modified. In Intel SGX there is a concept

[13] Software Guard Extensions

called Enclave. The establishment of a software "enclave" lies at the heart of SGX. The enclave is essentially a code and data-separated and encrypted sector. Because the enclave is only encrypted inside the CPU, it is protected against being read straight from the RAM. Enclave is used for both software security and hardware security which focuses on both memory and CPU [12].

4.4 Public Cloud vendors

4.4.1 AWS Lambda

Lambda, Amazon's Serverless Computing offering, adheres to the Shared Responsibility model in terms of security [2], relying on the customer to be responsible for their Code, Configuration, and IAM. As AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates, this shared responsibility model can help relieve the customer's operational burden. [5] AWS presented some principles that are visualised in figure 11.

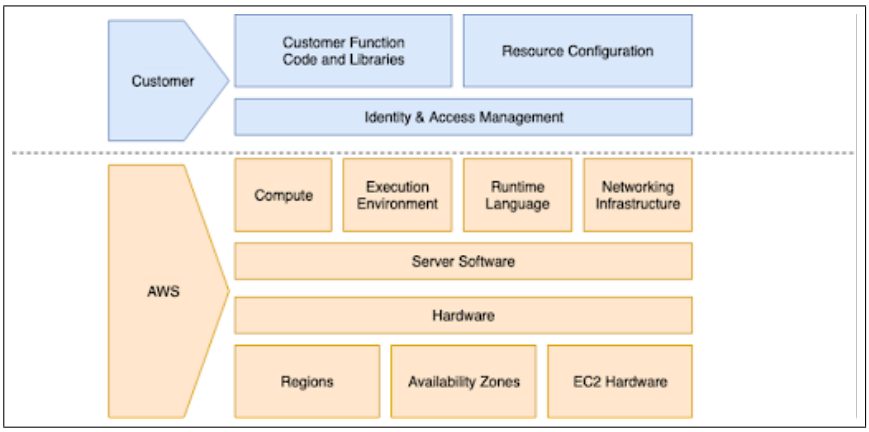


Figure 11: Shared responsibility model for AWS Lambda [5]

In a serverless context , AWS specifies these design principles: [3]
Speedy, simple, singular functions are required to be short,concise, single purpose and their environment may live up to their request lifecycle. Transactions are efficiently cost aware and thus faster executions are preferred [3].
Think concurrent requests, not total requests serverless applications take advantage of the concurrency model, and tradeoffs at the design level are evaluated based on concurrency [3].
Share nothing function runtimes environment and underlying infrastructure are short-lived, therefore local resources such as temporary storage is not guaranteed. State can be manipulated within a state machine execution lifecycle, and persistent storage is preferred for highly durable requirements [3].
Assume no hardware affinity Underlying infrastructure may change. Customers are asked to leverage code or dependencies that are hardware-agnostic as CPU flags, for example, may not be available consistently . They're also expected to orchestrate their application's workflow using state machines rather than functions. Chaining Lambda executions within the code to orchestrate the workflow of their app results in a monolithic and tightly coupled app. Instead, clients are expected to orchestrate transactions and communication flows using a state machine.[3].

Use events to trigger transactions Events such as writing a new Amazon S3 object or an update to a database allow for transaction execution in response to business functionalities. This asynchronous event behavior is often consumer agnostic and drives just-in-time processing to ensure lean service design. [3]

Design for failures and duplicates Operations triggered from requests/events must be idempotent as failures can occur and a given request/event can be delivered more than once.Developers are asked to include appropriate retries for downstream calls.[3]

Amazon also uses a variety of open-source and proprietary isolation technologies to protect Workers and execution environments and keeps a clear list of technologies used [4] allowing transparency into the actual hardware used for developers to debug any issues if they arise while also using community standard libraries and dependencies which are maintained by the open source community.

4.4.2 Microsoft Azure Functions

Azure functions is the FaaS service provided by Azure.This is service is designed to help customers build web APIs, respond to database changes, process IoT data streams, or managing message queues.To meet this need, Azure Functions provides "compute on-demand" in two significant ways.[18]

The first where Azure Functions allows the customer to implement their system's logic into readily available blocks of code. These code blocks are called "functions". Different functions can run anytime need to respond to critical events.[18]

Second, as requests increase, Azure Functions meets the demand with as many resources and function instances as necessary - but only while needed. As requests fall, any extra resources and application instances drop off automatically.[18]

The platform components of App Service, including Azure VMs, storage, network connections, web frameworks, management and integration features, are actively secured and hardened. App Service goes through vigorous compliance checks on a continuous basis to make sure that all communication and underlying infrastructure security.[19]The Developer documentation does provide easy access to guidelines on how customers could run more secure functions apps.Such as continuous logging of the service health and warn customers of any potential attacks.Customers are also required to use HTTPS due to the secuity of the underlying SSL/TLS protocol.Microsoft goes into detail on user practices to mitigate any potential attacks or loss of privacy.[19]

5 Advantages - Serverless Computing and FaaS security approaches

The most common way of holding a function in serverless computing is via Linux containers that are based on namespace and control group mechanisms. Each namespace can declare its own system resources which can only be accessed from inside its own namespace. This creates a security isolation method, since resources outside a namespace cannot access resources within that namespace. Another benefit of this approach is that the CPU, memory, network and disk usage is adapted to the necessary usage of resources within that namespace only, which establishes a security of resources taking up too much system resources. Despite some challenges with security, serverless computing also increases the security of applications. Function as a Service shifts the responsibility of the hygiene and maintenance of machines to the cloud providers who are more likely to update patches, so there is no risk of unpatched server vulnerability. This fact also to some degree establishes resistance against DDoS[14] attacks. In FaaS cloud environments, in case of a DDoS attack you can simply scale up to handle the load, due to the scalability functionality of FaaS [14, 17].

Based on papers we have read we have come to conclusion that followings are list of advantages of using serverless computing:

1. **Visibility to attackers:**As mentioned before, serverless applications are made as many number of functions in cloud which makes it so much easier for companies like google, amazon and etc. that work on security of their serverless applications, to infer internal workflow of application.
2. **Easier IAM:** Having multiple small functions make it easier to do. For security tools it is easier to define permission, access roles and security policies for each of small functions. This also results to more control over whole application by tools.
3. **Long-term attacks protection:** The fact that each function gets destroyed after doing the job, makes it so hard to put back doors in server because there is no resources assigned to function so attacker can not store any malware in application or do any manipulation.

[14]Distributed Denial-of Service

6 Discussion and Conclusions

In this review paper we categorized challenges of security in serverless computing and also introduced the cutting-edge tools to achieve maximum security in this area. Based on what we studied in this work, there are four different categories of security in serverless computing which we will discuss about their strengths and weaknesses and we introduce our ideas on how they could be improved. We also present our future ideas about how enthusiasts can do research further.

As mentioned in section 3 , one of the main challenges in serverless computing is data protection. We believe most researches in this area focus on 3 different categories like protecting data at REST API level, protecting flow of data inside the cloud and hardware-based approaches like Intel SGX. The strengths of these studies were that currently they cover most of challenges of protecting data in different scenarios. But still there are flaws in few topics like (1) hardware-based approaches are slow and are limited in user side and (2) solutions in this topic are not fully implemented for serverless environments, thus further researches on this matter required.

Another aspect of security was resource isolation which is one of hit topics for past few years and we can tell that it is well covered in research by presenting VM-based approaches and also providing concept of secure containers. We found out that VM-based approaches can bring better security but they are much slower because of their start-up time. That is why researches at first took the path of reducing the kernel code of OS on VM to make it lightweight. One of leading technologies in this area that we saw was Firecracker which was applied to AWS Lambda. secure container is more like industrial term which bring concept of lightweight VMs to reduce the start-up time with high level of isolation. Papers in this topic well covered the challenges of this area but we believe there are room for improvements including (1) making secure containers more reliable like VMs in term of security and isolation (2) containers still have some heavy computing that bring overhead and performance issues which can be tackled.

Security monitoring was another security aspect of serverless computing which requires more research. Papers in this topic focus on flow of data by putting dynamic tracing in functions. They also covered monitoring the traffic from outside of cloud to inside and for that matter they used REST API headers. But there are still room for more investigation on this matter like (1) these studies did not really focus on diagnosis rather they focus preventing and just observation. It would be much better to design a platform to diagnose and auto-fix procedure in monitoring (2) As mentioned in section 3.3 REST-based API headers was used to check for tags but no study focused on GraphQL queries for that matter. Tags could be used in every GraphQL query and mutation as a default configuration.

Last category that we discussed in security of serverless computing was security management. This category is all about managing capabilities, authentication and authorization. Works in this area focus on checking permission and access control over incoming request. Industry brought some definitions like IAM and PKI to manage permissions and keys. The strength of current researches are well definition of policies regarding access management but downside is that administration of that management system must infer the user needs and define policies and update them and it requires so much effort. One suggestion for improvement in this category is automation of this process. It is expected in the future that papers focus on automating inferring the user needs based on their behaviour and define some policies to meet their needs also maintaining and updating those policies over time based on changes on behaviours. Also in current paper there is no sign of quality checking of management. There is no specific tool for measuring quality of management in serverless computing.

To conclude, serverless computing comes with both security issues and security advantages. [14] highlights more how serverless computing and FaaS increase security by arguing that the maintenance and hygiene burden is shifted from developer to provider, and thus avoids non-updated patches.[14] also argues that DDoS attacks in serverless computing are handled better due to the ability of scaling up rather than a traditional approach. On the other hand, [26], discusses the matter of security based on cloud provider approaches. It focuses on security of containers and how Amazon and Google provide users with FireCracker and gVisor (respectively) to enhance the security of containers. Essentially, the complexity of serverless computing and FaaS comes with the necessity of strong management and process controls over organization's developments to keep their applications secure. Thus as mentioned in [17], organizations that cannot implement strong management and process controls of developments and environments, would be safer following classic security life-cycle

processes. In addition [17] discusses the security issues related to FaaS and its independence and complexity in detail.

Name	Work(Section)
Kimiya Ataiyan	2,3.4, 4.1, 4.2, 5
Behnam Bozorgi	1, 2.1, 3.1, 3.2, 3.3, 3.4, 4.2, 4.3, 5, 6
Abhilash Balaji	2.1, 2.2, 2.3 , 4.4 , 3.1

7 References

[1] Alexandru Agache et al. "Firecracker: Lightweight Virtualization for Serverless Applications". In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 419–434. ISBN: 978-1-939133-13-7. URL: https://www.usenix.org/conference/nsdi20/presentation/agache.

[2] *AWS Cloud Security: Shared Responsibility Model*. https://aws.amazon.com/compliance/shared-responsibility-model/. Accessed: 2022-05-23.

[3] *AWS, Architecting and Operating Lambda Functions*. https://docs.aws.amazon.com/whitepapers/latest/security-overview-aws-lambda/architecting-and-operating-lambda-functions.html. Accessed: 2022-05-23.

[4] *AWS, Lambda Isolation Technologies*. https://docs.aws.amazon.com/whitepapers/latest/security-overview-aws-lambda/lambda-isolation-technologies.html. Accessed: 2022-05-23.

[5] *AWS, The Shared Responsibility Model*. https://docs.aws.amazon.com/whitepapers/latest/security-overview-aws-lambda/the-shared-responsibility-model.html. Accessed: 2022-05-23.

[6] Ioana Baldini et al. "Serverless Computing: Current Trends and Open Problems". In: *Research Advances in Cloud Computing*. Ed. by Sanjay Chaudhary, Gaurav Somani, and Rajkumar Buyya. Singapore: Springer Singapore, 2017, pp. 1–20. ISBN: 978-981-10-5026-8. DOI: 10.1007/978-981-10-5026-8_1. URL: https://doi.org/10.1007/978-981-10-5026-8_1.

[7] Ioana Baldini et al. "Serverless Computing: Current Trends and Open Problems". In: *Research Advances in Cloud Computing*. Ed. by Sanjay Chaudhary, Gaurav Somani, and Rajkumar Buyya. Singapore: Springer Singapore, 2017, pp. 1–20. ISBN: 978-981-10-5026-8. DOI: 10.1007/978-981-10-5026-8_1. URL: https://doi.org/10.1007/978-981-10-5026-8_1.

[8] Pubali Datta et al. "Valve: Securing Function Workflows on Serverless Computing Platforms". In: *Proceedings of The Web Conference 2020*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 939–950. ISBN: 9781450370233. URL: https://doi.org/10.1145/3366423.3380173.

[9] Erwin van Eyk et al. "The SPEC Cloud Group's Research Vision on FaaS and Serverless Architectures". In: *Proceedings of the 2nd International Workshop on Serverless Computing*. WoSC '17. Las Vegas, Nevada: Association for Computing Machinery, 2017, pp. 1–4. ISBN: 9781450354349. DOI: 10.1145/3154847.3154848. URL: https://doi.org/10.1145/3154847.3154848.

[10] *Firecracker: Secure and fast microVMs for serverless computing*. https://firecracker-microvm.github.io/. Accessed: 2022-05-23.

[11] Puneet Gill, Werner Dietl, and Mahesh V Tripunitara. "Least-Privilege Calls to Amazon Web Services". In: *IEEE Transactions on Dependable and Secure Computing* (2022), pp. 1–1. DOI: 10.1109/TDSC.2022.3171740.

[12] *Intel SGX: Enclave*. https://www.intel.com/content/dam/develop/external/us/en/documents/overview-of-intel-sgx-enclave-637284.pdf. Accessed: 2022-05-23.

[13] Eric Jonas et al. *Cloud Programming Simplified: A Berkeley View on Serverless Computing*. 2019. DOI: 10.48550/ARXIV.1902.03383. URL: https://arxiv.org/abs/1902.03383.

[14] Philipp Leitner et al. “A mixed-method empirical study of Function-as-a-Service software development in industrial practice”. In: *Journal of Systems and Software* 149 (2019), pp. 340–359. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2018.12.013. URL: https://www.sciencedirect.com/science/article/pii/S0164121218302735.

[15] Yongkang Li et al. “Serverless Computing: State-of-the-Art, Challenges and Opportunities”. In: *IEEE Transactions on Services Computing* (2022), pp. 1–1. DOI: 10.1109/TSC.2022.3166553.

[16] *McAfee Labs Threats Report*. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-apr-2021.pdf. Accessed: 2022-05-24.

[17] J. Michener. “Security Issues With Functions as a Service”. In: *IT Professional* 22.05 (Sept. 2020), pp. 24–31. ISSN: 1941-045X. DOI: 10.1109/MITP.2019.2930049.

[18] Microsoft. *Introduction to Azure Functions*. https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview. Online; accessed 25 May 2022. 2022.

[19] Microsoft. *Securing Azure Functions*. https://docs.microsoft.com/en-us/azure/azure-functions/security-concepts?tabs=v4. Online; accessed 25 May 2022. 2022.

[20] Jussi Nupponen and Davide Taibi. “Serverless: What it Is, What to Do and What Not to Do”. In: Mar. 2020. DOI: 10.1109/ICSA-C50368.2020.00016.

[21] Rishabh Patil et al. “Serverless Computing and the Emergence of Function-as-a-Service”. In: *2021 International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT)*. 2021, pp. 764–769. DOI: 10.1109/RTEICT52294.2021.9573962.

[22] Weizhong Qiang, Zezhao Dong, and Hai Jin. “Se-Lambda: Securing Privacy-Sensitive Serverless Applications Using SGX Enclave”. In: *Security and Privacy in Communication Networks*. Ed. by Raheem Beyah et al. Cham: Springer International Publishing, 2018, pp. 451–470. ISBN: 978-3-030-01701-9. DOI: 10.1007/978-3-030-01701-9_25.

[23] Arnav Sankaran, Pubali Datta, and Adam Bates. “Workflow integration alleviates identity and access management in serverless computing”. In: *Annual Computer Security Applications Conference*. 2020, pp. 496–509.

[24] *Serverless Architecture Market by Service Type (Automation and Integration, Monitoring, API Management, Security, Analytics, and Design and Consulting), Deployment Model, Organization Size, Vertical, and Region - Global Forecast to 2025*. https://www.marketsandmarkets.com/Market-Reports/serverless-architecture-market-64917099.html. Accessed: 2022-05-23.

[25] *What is gVisor?* https://gvisor.dev/docs/. Accessed: 2022-05-23.

[26] Mingyu Wu, Zeyu Mi, and Yubin Xia. “A Survey on Serverless Computing and Its Implications for JointCloud Computing”. In: *2020 IEEE International Conference on Joint Cloud Computing*. 2020, pp. 94–101. DOI: 10.1109/JCC49151.2020.00023.