The Benefits and Pitfalls of Distributed Cloud-Based Software Architectures

Traian Svinti[1], Dillin Corbett[2], Mateusz Koltun[3] and Adam Craig[4]

[1,2,3,4] Dublin City University. Glasnevin, Dublin 9, Ireland
[1]traian.svinti2@mail.dcu.ie
[2]dillin.corbett4@mail.dcu.ie
[3]mateusz.koltun2@mail.dcu.ie
[4]adam.craig2@mail.dcu.ie

Abstract. Distributed Cloud-Based Software Architectures can sound like a complex idea to understand and an even harder concept to implement. The backend of these architectures can take different forms such as monoliths, micro-services and functions, where the latter types are becoming the most popular. This paper aims to reduce some of this complexity by providing the reader with the 'benefits and pitfalls' of distributed services in the cloud. Cloud computing pioneers, such as Amazon and Google, have made it possible to allow business owners to distribute their services on remote infrastructures.

The cloud aims to enhance user experience with its model. There are three key layers including necessary tooling that must be applied to thrive in this environment. The consequences of such technologies allow for the on-demand provisioning of software and hardware. Cloud services are provided to the end-user through smart devices with APIs or through web browsers.

Through the research conducted, we've educated ourselves in this field of study. Readers of this paper can gain a deeper impression regarding the nature of distribution in the cloud. Businesses, through the described relative merits, can determine whether to leverage this type of technology. Our recommendations suggest that the solution is with the cloud and that the positives do outweigh the negatives.

Keywords: Distributed, Cloud, Systems, Software Architecture, Virtualization(IaaS), Containerization(PaaS), Pros & Cons Synonyms

1.1 Introduction

In the past decade, major advancements have been made in IT architecture and software systems, transitioning from centralized systems to distributed and now cloud distributed systems. The cloud, or cloud computing, is no longer a 'buzzword'. Many IT companies have already or are in the process of migrating their software and data to cloud giants such as Amazon, Microsoft and IBM [1]. Current cloud architectures provide both storage and computation services for users and these are becoming a standard foundation to build software products. The cloud provides many benefits for consumers but is essentially the foundation of hosting a product/solution off-premise.

Prior to cloud computing, software architectures took the form of a localized monolithic systems which geographically resided in the same space. Each component

of the system is self-contained and interdependent. Once systems became larger and more complex with extra features being added, the limitations of the centralized architecture became apparent. Monoliths usually consist of one code base; front and back end, which can't be independently deployed. This resulted in increased downtime in order to release newer versions due to redeployment of the entire system [2].

Engineers began developing a new type of system where services would be decoupled and deployed in a virtualized or containerized environment. This gave way to flexibility, scalability and faster deployment to only the specific service. This incorporates the idea of horizontal scaling where instead of hardware being upgraded, a new node would be provisioned to handle the extra workload. The introduction of microprocessors in replacement of mainframes meant this system provided a better price/performance ratio without affecting how the end user interacted with the product. Microservices stem from this development and are considered distributed services [3].

The concept of distributed systems, such as a telephone networks, is to divide the overall functionality into smaller sub-processes. This can be achieved by using a group of computers that have a shared state and operate concurrently. To the end-user, the system appears as a single computer. Sub-processes focus on only one specific job and are executed independently to each other. Combined with cloud computing, they form one, single coherent system, whose function is to accomplish the same task as the original service [4].

Distributed cloud computing refers to the managing and provisioning of distributed resources in the cloud [5]. Many existing cloud-based software architectures are not distributed into separate components, meaning they consistently have unused resources. Idle CPU time and unused memory are examples [6]. With a distributed cloud-based architecture, these resources can be used to perform other tasks. The notion of distributed cloud-based architecture overcomes the downside of existing architectures. Over-provisioned resources aren't wasted, and with the distributed nature, it is possible to find a "cloud machine" geographically close to every user [6].

The remainder of this paper is organised as follows. Section III highlights the methodology. Section IV highlights the relative merits of distributed cloud-based software architectures. Section V considers the limitations of research and discusses future work. Section VI concludes this paper.

2 Related Literature

2.1 Methodology

This study involved the analysis of several peer-reviewed and non-peer reviewed articles, journals and blogs. We approached this in two ways; scoping and systematic review.

Scoping Review. Our knowledge in this field is limited and the area is very broad. Two team members were tasked with identifying major concepts/themes that are

associated with cloud computing and distributed computing. Terminology can be broad in IT and we wanted to make sure that we had as many relevant keywords mapped to our topic as possible. This meant adjusting and searching iteratively until the set was complete. This set was provided to the remaining team members who performed searches based on these keywords.

Systematic Literature Review. As a group, we formed our question that could lead us to a successful approach. “Can we identify major concepts and themes related to our topic that provides us with clarity to begin our granular search combinations and are these papers sufficiently related such that they contain the majority of the results set keywords?”.

Two team members in our group were tasked with obtaining relevant papers. Keywords, determined from our results set, were used in the scholar.google.com search engine. Information gathering commenced on the 11th of February to procure material that was as closely related as possible. We did not exclude any keyword from our initial search. To determine the relevance of the material that was returned, we analysed the topic title and summaries against our keywords.

If the paper was determined relevant, based on keyword match, they were included in an ordered list for a more in-depth review. The date and publisher were taken into account. This generated list was reviewed by our other two team members who originally identified the result set of keywords/concepts/themes. They accessed the material and proceeded to read the abstract, introduction and conclusion. If the material didn’t cover enough concepts, we dropped it from further review.

We noted, in some cases, that the google scholar search engine returned material from research hosted provider websites such as ResearchGate and ScienceDirect. These websites were used in conjunction with scholar.google.com in order to expand our knowledge base. Our total came to 40 resources of which 26 were peer-reviewed and 14 were not. We took care to ensure that no strict limits were applied, especially when it came to the date range. Our resources dated between 2002-2018, the majority of which was focused in the years 2013-2016 and we found this to be acceptable.

2.2 Inclusion / Exclusion Criteria

Inclusion. Naturally, our topic is quite broad and resulted in non-specific material. If through our initial searches on scholar.google.com, we were able to find our keywords in the topic title and summary, they were included in our review list. Throughout the course of this research, we did require assistance from our lecturer in the form of questions sent via email. These questions were then answered in class which helped us in our direction.

Exclusion. Specific papers, that did not cover enough general information from our keywords, were excluded. If no benefits or pitfalls were identified, they were excluded. We found that some material was quite detailed and advanced. Due to a limited amount of time, they were passed over. The majority of our non-peer reviewed material did not make it into our academic paper but we can’t omit the fact that they did provide us value and insight into the broader focus of our topic as well.

Unfortunately, the advanced search in google scholar did not provide the needed functionality to be more specific with our filtering.

2.3 Papers discovered and papers included

	Peer Reviewed	Non-Peer Reviewed
Number of papers identified	26	14
Number of papers included in the analysis	16	9
Total number of papers excluded	10	5

3 Analysis

3.1 Cloud Computing

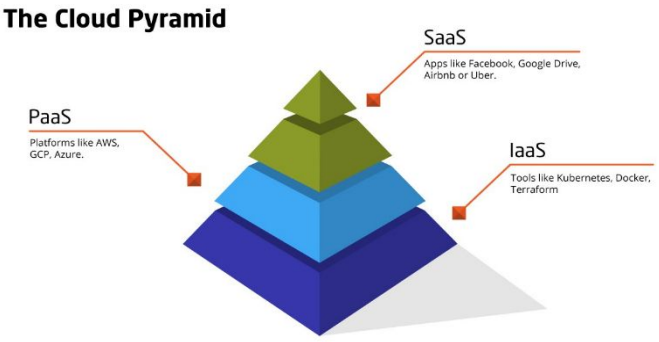


Figure 1. The Cloud Pyramid [19].

Figure 1 depicts the cloud structure. This allows identification of each layer that makes up the cloud. Below, each section is analysed in a bottom-up style.

IaaS - Infrastructure as a service. The computing architecture and infrastructure is offered in this layer through a virtual environment. No physical computer is involved or needed to be maintained by the customer. Virtualized instances can belong in multiple geographical regions. This allows for great flexibility. Sysadmins or DevOps maintain this layer to prevent issues with the data, application, runtime or other middleware. Examples: Docker, Jenkins. However, comparing with other XaaS, IaaS is more difficult to maintain. It requires an experienced engineer who configures the virtual machines to work efficiently and securely [7].

PaaS - Platform as a service. PaaS is a programming language execution environment. Essentially encapsulating the environment where developers can build, compile and run their programs. They manage data and applications resources. There is no need to worry about the underlying infrastructure. It's scalable, allows for faster market delivery and simplifies the deployment for web-based applications. A

downside of this is that developers are limited to provider's tools, languages and migration issues such as vendor lock-in can occur [8].

SaaS - Software as a service. SaaS offers on-demand pay per use of application software to end users delivered over the internet. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities. The service is platform independent and does not require the installation of the application on the user's device. The computing resources are managed by the vendor of the service. Usually accessible via a web browser but can be accessed via lightweight client applications. Since SaaS software is web-hosted, one can't use these applications without an Internet connection. If the Internet service goes down or if mobile workers are in an Internet dead zone, one won't have access to the software or data [9].

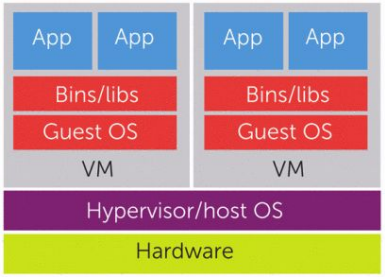
Now that each layer has been dissected in the cloud pyramid, the paper introduces the topic of virtualization and containerization. They play a significant role in distribution but are crucially more important and applicable, when it comes to the cloud.

3.2 Virtualization/Containerization

In a monolith architecture, the entire backend is deployed in one place. This makes it bulky and difficult to scale. In contrast, microservices take advantage of each layer of the Cloud Pyramid, alleviating the issues of a monolith.

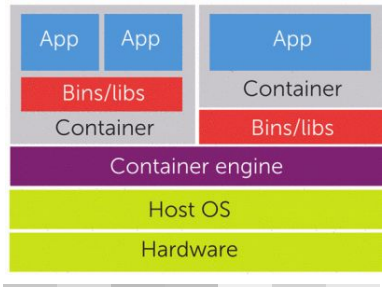
Virtualization of microservices is the concept of taking a microservice and running it in its own independent container. This container is built from an image. This image describes the structure, required dependencies to run services and the protocols used for communication. The benefits of such virtualization touch on areas such as scaling, reusability, availability and ease of refactoring. Each of these benefits is a by-product of the main point addressed by virtualization and deployment [10].

Virtualization is implemented by importing a guest operating system on top of a host operating system. This technique allows developers to run multiple operating systems in different virtual machines. This eliminated the need for extra hardware resources. Described are the following advantages of virtualization. Multiple operating systems (OS) can run on the same machine. Maintenance and recovery are made painless in case of failure conditions. The total cost of ownership was also less due to the reduced need for infrastructure [11].



In Figure 2 [10], there is a host operating system on which there are two guest operating systems running; these are the virtual machines. Virtualization, although a revelation, has some shortcomings. Running multiple virtual machines in the same host operating system leads to performance degradation and unstable performance.

This is due to the guest OS running on top of the host OS, which will have its own kernel, set of libraries and dependencies. This takes up a large chunk of system resources, i.e. hard disk, processor and especially RAM. Another problem with virtual machines is that it takes almost a minute to boot-up [10] [12]. This is very critical in the case of real-time applications. These drawbacks led to the emergence of a new technique called Containerization.



Containerization provides cloud providers with the flexibility to relocate, instantiate and optimize hardware-based resources with minor impact on performance. Containerization is widely discussed as a lightweight virtualization solution. As it can be seen in the Figure 3 [10], each container only contains its libraries, binaries and a specific application. There is no guest OS because the parent host's OS and kernel are used instead [12]. Apart from exhibiting benefits over traditional virtual machines in the cloud, containers are especially relevant for PaaS clouds, which use them to manage and orchestrate applications via software packaging mechanisms.

Discussed are the requirements that arise from having to facilitate applications through distributed multi-cloud platforms. Containerization is the technique of bringing virtualization to the operating system level. While virtualization brings abstraction to the hardware, containerization brings abstraction to the operating system [10]. Containerization is also a type of virtualization and is more efficient due to the absence of a guest OS and utilizes a host's operating system, kernel, share relevant libraries & resources as and when needed, unlike virtual machines.

Application specific binaries and libraries of containers run on the host kernel, which makes processing and execution very fast. Even booting-up a container takes only a fraction of a second. Because all the containers share, host's operating system and holds only the application related binaries and libraries. They are lightweight and faster than virtual machines [12].

Next are the overall key 'benefits and pitfalls' that we uncovered. that highlight what coupling can be expected with distribution in the cloud.

3.3 Benefits of Distributed Cloud Computing

Deploying and running services in the cloud has many benefits, for which companies decide to switch from old fashion, bare metal infrastructure to modern architecture that is cloud computing. One of the most important advantages that cloud services offer is high availability. Businesses thrive to keep their services running with minimal downtime, making their service as reliable as possible [13].

Cloud providers use appropriate engineering methods for calculating the availability of their infrastructure. Probability Risk Assessment (PRA), is one of these methods. The same method that is used in aircraft, defence, insurance and the nuclear power industry [13]. MTechnology, a leading firm in the field of risk assessment has conducted tests in one of data centres. The results had shown that from the power

supply perspective, traditional system design in practice can deliver four 9s (99.99%) availability [13]. This score is equivalent to 4.38 minutes of downtime per month. However, this is a limitation of a power source in one data centre. Normally, well-distributed systems are deployed in more than one data centre by using availability zones as isolated locations. This approach allows to surpassed the four 9s limitation.

The availability of a service depends on both software and hardware. To achieve a high score in availability, redundancy must be achieved in both. Developers, however, have more control over the former. Clustering a service is one way of improving availability from a software point of view. These type of solutions can become sophisticated very quickly and can be limited by hardware and other resources. Cloud computing allows developers to exceed these limitations. The ability to create as many virtual machines as desired in various locations makes the aim of six 9s (2.6s of downtime monthly) achievable, both in practice and theory [14].

Cloud computing services present heavy reliance on resource infrastructure and network availability at all times [15]. The redundancy is applied to every component which has a single point of failure. In networking, these components are switches and aggregators on a physical level of the OSI model. On top of that, cloud providers would normally use multiple Internet Service Providers (ISP) to increase the reliability of their services.

To summarise, high availability is only obtainable in the cloud as long as the resources which cloud providers offer are used correctly. With that said, components of a service should be distributed across multiple availability zones to gain full potential of reliability [16]. In a way that data replication and service redundancy use available resources in two independent locations.

Scalability is yet another factor which every engineer should take into consideration when designing the architecture of a system. It refers to the ability of how well a system can handle the growth and additional workload which comes with it [18]. A distributed system is a type of architecture which contains scalability as a feature in its own definition. Scalability can become a serious issue in software applications. Making sure a service is scalable, from a software point of view is a difficult task on its own. A distributed system can get quite complex in its design, especially, while trying to keep its components truly parallel.

A distributed system is no longer limited to being provisioned with on-premise hardware. Cloud services are flexible, they allow a user to control how and when resources in their system are used. The user decides in which geographical location they want their system to scale and to what size. This is a huge advantage from the operational point of view. As any manual labour, when it comes to hardware and its maintenance, is taken away from a user perspective.

However, once a service is well designed, implemented and ready to scale, the infrastructure and resources become a limitation very quickly. Cloud computing is not necessarily an ultimate solution to the problem. Nonetheless, it allows one to scale service in a controlled and an automated manner [17]. The virtualization technologies can dynamically provision a server on-demand. Then it can be ready to handle the chunk of the service workload given to it.

A user has full control, when an additional node is added to a clustered service or when it should be deleted from it [17]. Scaling factors can be adjusted to fit the requirements of the service. Auto-scale groups are smart enough to determine if a service needs additional resources. This is based on CPU usage, network congestion or even a time of a day. Distributed systems benefit more from horizontal scaling and cloud services provide right tools to do it well. Although, virtual scaling is also possible. Less powerful virtual machines can be substitute with one that has more resources. Both of these strategies follow the same on-demand model which is specified by the user.

As mentioned previously, a distributed system is a set of components that work with each other in collaborative fashion. In order to keep the workflow consistent between the components, well-defined interfaces are crucial. A specific protocol of communication for sending and receiving messages must be agreed on. This strategy ensures that data flow is standardized across the whole system which allows for further service integration and expansion [11].

Unfortunately, this introduces some constraints, each additional component added to the system increases the required amount of messages to be sent. This can become a problem very quickly and limit the performance of a system. Systems that are distributed on already congested network are especially endangered here. Another factor limiting the communication is latency which is directly correlated with location where components of a systems are hosted at.

However, having a system in the cloud, doesn't necessarily eliminate these issues, but greatly reduces the impact of them. Cloud services run on a dedicated network, and distributed systems highly benefit from this. By provisioning Virtual Private Networks (VPN), a user has full control of the bandwidth limitation. Cloud providers also make use of edge server architecture to improve the connection to their services from locations that are located far from main data centres.

A major advantage of decoupling services in the cloud is the possibility of running batch jobs. This makes it a straightforward process to use 1000 servers to accomplish a task. Efficiency of the target task is increased to 1/1000 times that of a single server doing the same job [21]. Refactoring of services that need good response times for queries is done by 'farming' out the work to virtual machines on edge servers [21]. This process not only increases the efficiency in completing the task but also gives way to horizontal scaling on-demand. More requests are handled at once and a new instance can be automatically generated to offset load from other containers.

When it comes to the cost associated with running decoupled microservices in the cloud, in a long term, compares much better than with having a proprietary system. The initial costs of migration may be high but running costs are greatly decreased.

A benefit of this is that there is no longer a need for an IT department to look after, investigate and resolve infrastructure related issues. This greatly reduces the complexity and cost of having a proprietary infrastructure. The resources used for this work can now instead be used in a more beneficial role. Companies now have better control over how much they want to spend regarding the infrastructure. Cost limits are put in place for services such as auto-scaling. This is due to cloud providers using a pricing scheme according to incurred resource consumption.

For example, Amazon EC2 provides a virtual machine with a single CPU core at a price of $0.095 per hour. This pay-as-you-go model lets users utilize a public cloud at a fraction of the cost of owning a dedicated private one while allowing providers to profit by serving a large number of users [20]. Case studies from these cloud providers indicate that a variety of applications have been deployed in the cloud, such as storage backup, e-commerce and high-performance computing. For a provider, it is a non-trivial task to define a uniform pricing scheme for such a diverse set of applications [20].

3.4 **Pitfalls of Distributed Cloud Computing**

This paper discussed multiple benefits of distributing a system in the cloud, outlined below are the drawbacks.

Going from a monolith structure to distributed microservices does not mean that once the migration process is complete that everything will be simpler; the complexity simply shifts. Many factors play into the complexity of the migration in itself such as time, resources, and initial costs [23]. Even with Migration Patterns being available, choosing the correct pattern to follow can be time consuming. If not done correctly, re-evaluation factors would need to be considered. This sort of planning can be difficult for companies who have adopted an AGILE planning approach. Many resources would need to be put into place to create the system and also test it.

Companies may not have the revenue to outsource this work and would also not have the headcount to move people over that are currently working on the service itself. Communication can be a major issue in decoupled services, especially in the cloud. If services are not smartly distributed and automatic scaling is not being utilized, services can become unresponsive under heavy load. If there is a Distributed Denial of Service (DDOS) attack, service farms would not be able to handle the overall load. They could fail to execute any of the requests, whether valid or not.

The above issue can be avoided by utilizing cloud providers that offer automatic scaling. This will create a new instance of the service and load balance the requests. Meaning that services will not fail. In the event of a DDOS attack, this can be rather costly if new services keep being created to handle the load [21].

Another issue that a badly distributed system, with a microservices backend architecture, may run into is data expectations. If a service is expecting a specific value type to be received and it receives a different value, this will, of course, result in an error. This can be hard to trace due to the diversity in the locations of data transmission. Observability concerns can be resolved through a combination of metrics, logging and tracing. Although, this is a non-trivial solution.

Services being distributed have many major advantages, but not without their pitfalls; Privacy being a major problem with a cloud-based distributed system. The reason for this is even though cloud providers usually reside in the ‘tech giants’ such as Google, Amazon and Microsoft; the company using the cloud service does not have overall visibility of the system as a whole. In the cloud providers network, there are many entry and exit points, these cannot all be monitored by a user of the service. If any of these are exploited, information may be leaked [22]. Cloud providers

currently lack an implementation for application owners to effectively collect and monitor weak points in the system.

Another privacy concern also involves the communication of the distributed microservices. To be able to communicate effectively and efficiently, they are completely open to each other's data requests; validation is not generally implemented between internal, communicating services. This entails that if one microservice is compromised, the system as a whole also is. As a simple example, this could be something as small as SQL injection where information is pulled from other microservices such as databases through a form input such as a search bar. Cloud providers lack the implementation of confined trust between microservices [22].

The above pitfalls resulted in the belief that Cloud Computing (CC) is still in an early stage of its life cycle and thus, lacks standardization. Each CC provider proposes and creates their own unique interfaces, resources and services. This leads the user of the provider to be confined to that specific digital ecosystem, making it difficult to migrate or expand to other providers [8]. Providers knowingly implement their systems in a unique way to ensure complexity in the migration away from their service to a possible competitor. All deployed services now have complete dependence on the health of the cloud providers network and hardware. This outlook deters organizations from deploying their distributed services in the cloud.

4 Discussion

Throughout this research paper, the domain of distributed cloud-based software was analysed. Having then presented a detailed examination and breakdown of what a distributed cloud-based system entails. Advantages and disadvantages of such systems were subsequently discussed. The findings and limitations of the research carried out will now be reviewed.

4.1 Limitations of Research

This research paper is the work of only four students, a dramatically small number when the size of content and number of existing publications on this topic and related topics is to be considered. With that being said, it is almost certain that the content and conclusions of the findings could possibly be an outcome of chance. Further literature could exist containing contrasting information on the same topics. Such information could propose different facts and advantages, which essentially if reviewed could result in the formation of a different paper.

There is an indefinite lack of research experience between the group in this field, with quite a lot ambiguity in terminology occurring when exploring low-level detailed journals. A second shortcoming of having limited research experience, which was undoubtedly exploited, was having no knowledge of the authors of such papers. This implied that the choice of papers to reference was based solely on content and citations. An author's experience or credibility in a specific software field was disregarded.

Another limitation encountered was the time frame in which this paper had to be completed.

It must be noted that if no deadline were to exist, having extra time for research, information retrieval, writing, refactoring and references would have undoubtedly improved the quality of this paper.

4.2 Future Work

In sight of the execution of the research and formation of this paper, considering the above limitations encountered, perhaps a good future approach would be as follows: - First, research the discussion and limitation sections of studies to develop a general awareness of such issues raised prior to beginning the research. Having performed this action, it is plausible we would be able to overcome one or more limitations by considering these factors in the methodology. For example, more factual information could be obtained. A portion of our time could be spent studying author credibility and experience on the topic of distributed cloud-based systems.

As mentioned throughout the paper, there are existing negative's in moving to a distributed cloud-based architecture. What has not been mentioned are the predicted fast approaching problems with the current cloud design. As estimated by Cisco Global Cloud Index, by 2019, the data produced by people and machines will grow to 500 zettabytes, however, the global data centre IP traffic will only reach 10.4 zettabytes by that time [24].

It has also been estimated that there will be 50 billion devices connected to the Internet by 2020, as predicted by Cisco Internet Business Solutions Group. Astonishingly this calculates to around 6.25 billion devices for every living person. Many of these applications and services will require a very short response time while producing a large quantity of data which could be a heavy load for networks. Cloud computing is not efficient enough to support these applications [24].

Such concerns have brought rise to the concept of 'edge computing'. Edge computing is a new idea in which substantial computing and storage resources (referred to as cloudlets) such as microdata centres or fog nodes, are placed at the internet's edge in close proximity to mobile devices or sensors [25].

Future research in edge computing could bring rise to possible solutions for the upcoming strain which is to be placed on the current distributed cloud architecture.

4.3 Conclusions

In this thorough research paper, a lot of valuable information regarding the distribution of software in the cloud was found. A major topic that was discussed is the infrastructure of the cloud and the services that it provides (cloud pyramid). Each layer is expressed in detail from IaaS, PaaS and SaaS. Virtualization and containerization were discussed and how these compare and contrast to each other. Comprehensive benefits and pitfalls were then laid out in a structured manner.

Benefits discussed include high availability in regards to service reliability; the simplicity in upscaling a service based on load or need; elasticity and flexibility in

having full control over how and where cloud resources are used; improved communication between system's components on dedicated, virtual private networks; how the cost greatly decreases with running services only when they are needed and how responsive edge servers are at handling requests. Each benefit is a key point of the power that comes with distributing software in the cloud.

Pitfalls discussed include how privacy is impacted as information is no longer stored on proprietary hardware; load and incorrect data rights for microservices can break the system if not correctly implemented; observability becomes an issue for the user of the cloud service as one cannot see how information reaches another service; there is complete trust between containers which means one being compromised would give access to all services; the complexity and cost of migration and once the user is tied down to a cloud provider it would be difficult to move to another.

From the investigation and analysis, one can see many more pitfalls than benefits, but by analysing the data, it is believed that the benefits out way the pitfalls, concluding that distributing a software architecture in the cloud is an astute concept for any scaling business owner.

5 References

[1] Gai, K., Li, S , *Towards Cloud Computing: A Literature Review on Cloud Computing and its Development Trends*, 2012 Fourth International Conference on Multimedia Information Networking and Security, January 2013, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6405648

[2] Namiot, D., Sneps-Sneppe, M. , *On Micro-services Architecture*, International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no. 9, September 2014, https://www.researchgate.net/publication/265292970_On_Micro-services_Architecture

[3] N/A, *Cloud Computing vs. Distributed Computing*, dezyre.com, April 2015. https://www.dezyre.com/article/cloud-computing-vs-distributed-computing/94

[4] Kozlovski, S., *A Thorough Introduction to Distributed Systems*, April 2018. https://medium.freecodecamp.org/a-thorough-introduction-to-distributed-systems-3b91562c9b3c

[5] Kaur, P. , Rani, R., *Distributed and Cloud Computing Architecture*, Imperial Journal of Interdisciplinary Research (IJIR) Vol-3, Issue-2, 2017 ,https://www.onlinejournal.in/IJIRV3I2/190.pdf

[6] Chan-Tin, E.; Khethavath, P., Liu, H., Thomas, J. , *Introducing a Distributed Cloud Architecture with Efficient Resource Discovery and Optimal Resource Allocation*, 2013 IEEE Ninth World Congress on Services, November 2013 https://ieeexplore.ieee.org/document/6655725

[7] Mezgárab, I., Rauschecker, U. , *The challenge of networked enterprises for cloud computing interoperability*, Computers in Industry Volume 65, Issue 4, May

2014, Pages 657-674, https://www.sciencedirect.com/science/article/pii/S0166361514000335

[8] Opara-Martins, J., Sahandi, R., Tian, F.,, *Critical review of vendor lock-in and its impact on adoption of cloud computing*, IEEE Cloud Computing, November 2014, https://ieeexplore.ieee.org/abstract/document/7009018

[9] Fatih Erkoç, M., Bahadir Kert, S. , *Cloud Computing For Distributed University Campus: A Prototype Suggestion*, International Conference The Future of Education, June 2015, https://www.researchgate.net/publication/228812233_Cloud_Computing_For_Distributed_University_Campus_A_Prototype_Suggestion

[10] Pahl, C., *Containerization and the PaaS Cloud*, IEEE Cloud Computing,IEEE Cloud Computing (Volume: 2 , Issue: 3 , May-June 2015), https://ieeexplore.ieee.org/abstract/document/7158965

[11] Celesti, A., Chen, L., Fazio, M., Liu, C., Ranjan, R., Villari, M., *Open Issues in Scheduling Microservices in the Cloud*, IEEE Cloud Computing (Volume: 3 , Issue: 5 , Sept.-Oct. 2016) https://ieeexplore.ieee.org/abstract/document/7742215

[12] Dua, R., Kakadia, D., Reddy Raja, A., *Virtualization vs Containerization to Support PaaS*, 2014 IEEE International Conference on Cloud Engineering March 2014, https://ieeexplore.ieee.org/abstract/document/6903537,

[13] Robertson, C., Romm, J., *Data centers, power, and pollution prevention*, June 2002, https://www.researchgate.net/publication/30482400_Data_centers_power_and_pollution_prevention_design_for_business_and_environmental_advantage

[14] S. Srinivasan, *Cloud Computing Basics* (46), Springer Briefs in Electrical and Computer Engineering, May 2014

[15] Dimitrios Zissis, Dimitrios Lekkas, *Addressing cloud computing security issues* ,*Future Generation Computer Systems*,(Volume 28, Issue 3, March 2012, Pages 583-592) https://www.sciencedirect.com/science/article/pii/S0167739X10002554

[16] N/A, AWS Documentation/Amazon EC2/User Guide for Linux Instances /*What Is Amazon EC2?*, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.htm

[17] N/A , AWS Auto Scaling, *Application scaling to optimize performance and costs*, https://aws.amazon.com/autoscaling

[18] Phil, Linux Academy Blog; *What is scalability in cloud computing?* , July 2018, https://linuxacademy.com/blog/cloud/scalability-cloud-computing

[19] Fedak, V. , *What is the cloud pyramid: the layers of DevOps services ?* , May 2018, https://medium.com/@FedakV/what-is-the-cloud-pyramid-the-layers-of-devops-services-730ac137e8b8 ,

[20] Chen, R., He, B., Jing, Q., Qian, Z., Wang, H., Zhout, L., *Distributed Systems Meet Economics: Pricing in the Cloud,* June 2010,

https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/WangH.pdf,

[21] Sharma, R., *Cloud Computing; New Cutting Edge to Business*, Redefining the IT Business, INTERNATIONAL JOURNAL OF COMPUTER APPLICATION (ISSUE 2, VOLUME 2) APRIL 2012, http://rspublication.com/ijca/april%2012%20pdf/12.pdf

[22] Jaeger, T., Nanda, S., Sun, Y., *Security-as-a-Service for Microservices-Based Cloud Applications*,2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom) Dec 2015, https://ieeexplore.ieee.org/abstract/document/7396137

[23] Chinenyeze, S., Pahl, C., Jamshidi, P., Liu, X., *Cloud Migration Patterns: A Multi-Cloud Service Architecture Perspective*, 2015, http://doras.dcu.ie/20331/1/WESOA-CloudMigrationPatterns-Final.pdf

[24] Cao, J., Li, Y., Shi, W., Xu, L., Zhang, Q., *Edge Computing: Vision and Challenges*, IEEE Internet of Things Journal (Volume: 3 , Issue: 5 , Oct. 2016), https://ieeexplore.ieee.org/abstract/document/7488250

[25] Satyanarayanan, M., *The Emergence of Edge Computing,* Computer (Volume: 50 , Issue: 1 , Jan. 2017), https://ieeexplore.ieee.org/abstract/document/7807196