# Microservice dynamic resource provision for small and medium-sized enterprises

Adam Irufaan
*School of Computing*
*Asia Pacific University of Technology and Innovation (APU)*
Kuala Lumpur, Malaysia
TP020295@mail.apu.edu.my

Hemalata Vasudavan
*School of Computing*
*Asia Pacific University of Technology and Innovation (APU)*
Kuala Lumpur, Malaysia
hemalata@staffemail.apu.edu.my

Palvinderjit Kaur Harnek Singh
*School of Computing*
*Asia Pacific University of Technology and Innovation (APU)*
Kuala Lumpur, Malaysia
palvin@staffemail.apu.edu.my

***Abstract*— For small and medium-sized enterprises (SME) it is important to quickly adapt to the ever-changing business environment while also keeping costs low. By adopting a microservice architecture based on containers, SME's can cut costs on the infrastructure cost and also increase the performance of the services provided by the information technology infrastructure. Moreover, microservices can further help reduce the cost within an organization by reducing the technical debt which can be increased by maintaining a monolith system. Further maintenance cost can be cut through the help of virtualization technologies which can provision resources dynamically for microservices. As SME's run on a smaller budget compared to bigger organizations, it is important for an SME to be able to take full advantages with these technologies. For that reason, a model for converting monolith systems to microservices which support dynamic resource provisioning is required.***

***Key words - microservices, virtualization, container, docker, SME***

## I. INTRODUCTION

Microservices are becoming more popular compared to monoliths. Thanks to microservices it has become easier to manage the load that the servers that host these systems are under. In microservices, different services are under different loads at any given time and to cater to this issue valuable time is spent on allocating resources for the services that are under load. By automating the resource allocating process, the workload can be reduced. Furthermore, by scaling down services when they are under low load, the financial burden can be reduced by lowering electricity costs as well. In this literature review, the advantages of running microservices for small and medium-sized enterprises (SME) will be highlighted along with how microservices communicate with each other. Furthermore, the advantages of using a microservice architecture instead of a monolith for an SME will be identified as well. As microservices grow the need for scaling becomes more pressing. Virtualization plays an immense role in helping microservices scale. This literature review will also highlight the methods of virtualization along with the advantages and the disadvantages that they come with. After identifying the methods of virtualization, the most commonly used and the most cost-effective method will be identified. Lastly, this literature review will look into scaling. Previously used scaling methods and the outcomes that they had will also be focused on. This can help an SME to further consider to use microservices in a containerized environment to better manage and scale the systems that they manage.

## II. MICROSERVICES IN SME'S

The microservice architecture is a distributed application where all the modules within it provide services to each other [1]. In a microservice architecture, microservices are small, independently deployed and autonomous which has a single purpose which is clearly defined [2]. One of the primary differences between traditional service-oriented architecture and a microservice architecture is that microservices communicate between each other using standards such as HTTP, JSON and REST instead of using an enterprise service bus [1]. Furthermore, service-oriented architectures are not able to scale as efficiently as well. This is because service-oriented architecture generally exposes monolith services and to scale a monolith, the whole system needs to be scaled instead of a single module that needs to be scaled in microservices.

### *A. Maintainability and fault tolerance*

One of the main reasons why microservices are favored is due to the maintainability which it offers. The maintainability of microservices can be attributed to the fact that microservices are decoupled which reduces its complexity [3]. Furthermore, while development in a monolith system may require the whole system to be redeployed for updates [4] the microservices architecture allows independent deployment as well. The ability to be deployed independently provides the system with various advantages. These microservices can be developed in any programming language which can ease the development process and be deployed on any hardware that best suits the needs of the microservices [2]. Moreover, microservices can scale independently from other services and is fault-tolerant [2]. If microservices can isolate failure and do not cascade, the system can still carry on working unlike in a monolith system where a failure can stop the whole system from working.

Even though microservices provide many advantages compared to monolith systems some disadvantages are also present. As microservices are distributed systems, it is possible to have more complexities in microservices compared to monolith systems [1]. Furthermore, the skill of the implementation team plays a major role in the efficiency of the microservice architecture [1]. This suggests that even when implementing a microservice architecture in an SME, the SME must carefully invest in the development team to produce an efficient microservice which also is cost-efficient in the long run. With less investment in the skills of the development team, an organization can implement an efficient

monolith system while a team with good skill sets is required to produce an efficient microservice architecture [1]. This means that to produce a microservice architecture, investment is required in the skill levels of the development team.

### *B. Agility*

Apart from the technical benefits, an organization can generate performance-based benefits as well with the help of choosing an easily maintainable infrastructure [5] which microservices can provide [6]. One of the reasons that such technologies as microservices provide advantages to business is due to the agility that it provides [1]. For a business, the agility is how fast they can detect and respond to a threat or an opportunity with ease, speed and dexterity [6]. With greater agility, an organization can update the portfolio of IT applications much faster [6].

Large monolith systems can slow down how an organization can respond to a change in the business environment [1]. As a system age, it will both grow and become more complex. Unlike in monolith systems, microservice architectures provide agility because microservices allow independent development and modifications within a microservice unlike in monolith systems where the whole system be required to change for deployment to take place after a development [1].

### *C. Technical Debt*

Different organizations are opting for microservice architecture over monoliths to facilitate easier software maintainability or even to improve quality of the systems, furthermore with the help of microservices organizations can reduce the technical debt (TD) in organizations which in turn reduce the maintenance cost as well [2, 6]. TD can be described as the debt which is introduced due to speeding up of development phase of software which can increase the deficiencies which in return can increase the maintenance cost of the software [2]. TD growth in microservices was observed to be lower than the growth in monolith systems [2]. A higher TD can cause the organization to have a higher operational expense due to the requirements in hiring more experienced developers or even reduce the performance of the developers.

## III. VIRTUALIZATION

The efficiency of resource utilization is one of the biggest concerns when deploying applications to the cloud [7]. Microservices can be scaled using elastic computing. For elastic computing, physical resources assigned are adjusted to each of the deployed applications [4]. There are two methods for scaling which are vertical and horizontal scaling. In vertical scaling, more virtual machines (VM) are added to increase the availability of resources while horizontal scaling is where more resources are allocated to a deployed VM [4]. Even though both of the methods can be used to scale, horizontal scaling can take a much longer time compared to vertical scaling [4]. Even though VM's are traditionally used for elastic computing, the use of containers is now becoming more popular due to the advantages that they offer.

### *A. Virtual Machines and Containers*

Even though previously VM's were the most popular technology for virtualization, container-based virtualization methods are now becoming more accepted due to containers being more lightweight [8] along with other benefits that they offer. With the use of containers based virtualizations compared to VM, scaling becomes more responsive as the resource overhead is reduced [6, 9]. Furthermore, containers offer better scalability and elasticity in the development environment [4] making the use of containers more favorable. Apart from scaling, containers provide other benefits in dependency management. By using containerization technologies such as Docker, developers can easily manage consistent development and deployment environments due to the ease of dependency management [3, 6, 10]. Apart from requiring fewer resources, containers can even startup faster than a traditional VM which can take around a minute to startup [3, 4]. The startup time when using containers compared to a VM means horizontal scaling can also be done faster. The reason why horizontal scaling is slower on a VM is that they take a longer time compared to containers on startup [4]. The differences in the isolation and how virtualization is done illustrated below in Fig 1.

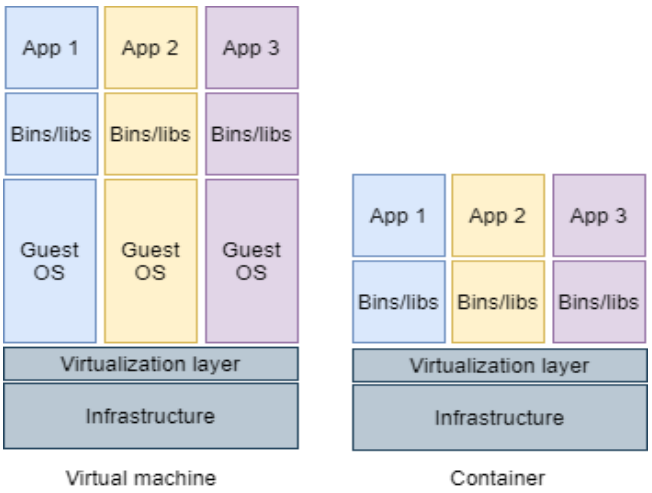


Fig. 1. Comparison of virtualization between virtual machines and containers

As seen in Fig 1, for virtualization each instance requires an operating system to be installed which will require additional space compared to containers. In the case of containers, the required operating system files are shared throughout the containers from the infrastructure layer where the operating system resides. This is one of the reasons why containers are lightweight as only the files and libraries which are not provided by the operating system is included in the container [11]. Moreover, containers offer independent execution environments while also providing isolated file systems as well [11]. Furthermore, along with the advantages of being lightweight, the deployment of containers has also become much easier with the emergence of container management tools such as Kubernetes [11].

Both virtualization methods offer different isolation as well. While a VM can offer isolation at the hardware abstraction level, containers offer both operating system level abstraction and isolation [4, 11]. The advantage in an operating system level isolation is that the hardware resources can be shared amongst other containers within the physical machine [4] which allows more resources to be available in the container-based virtualization method. On the other hand, in a VM fewer resources are available as each VM requires an operating system to be installed before they can be used [4].

When considering more relevant technologies for containers such as Docker which can help in both the development and deployment processes, the advantages of using containers over a solution with VM for microservices become even more clear. An algorithm that takes the features of Docker into consideration named EPTA compared to VM based scaling solutions as it showed improvements in physical machine startup time, supporting library cost and the communication cost as well [4]. The library support cost was reduced as containers support operating system isolation instead of hardware isolation. The notion of containers having an even greater efficiency for scaling when using docker scaling strategies is further supported by Guan *et al* [4] as tests carried out shows efficiency in scaling with containers with docker compared to VM.

### *B. Using application-level matrices for scaling*

Moreover, containers do allow microservices to be deployed to Software-as-a-Service (SaaS) vendors. When SaaS vendors handle the microservices it introduces some caveats. As business activities such as promotions or any other such activity can drive more traffic to a microservice which can increase the load. This load is not known beforehand for the SaaS vendors. Such increases in traffic for microservices can create uncertainty in the load for the SaaS vendors, and if the required information is not provided the vendors are not able to scale the microservices properly [6]. Another issue is the timeliness and accuracy of autoscaling which can arise, this can occur due to the limited rule-based autoscaling functionalities from conventional container orchestration platforms [6]. Ideally, auto-scaling should be based on not only the resource metrics but also with the application-level metrics as well [6, 12]. By using application-level metrics as well uncertainty in load can be better handled. Zheng et al [6] observed a 60% reduction in cost when considering these issues and by utilizing application-level metrics as well when auto-scaling.

### *C. Security*

Apart from other benefits virtualization provides, virtualization also helps businesses to better facilitate compliance to IT regulations and management through security benefits as well [13]. As well as helping organizations to reduce the number of servers required to run the systems due to lower resource utilization, it also helps to reduce the power usage and cooling facilities as well [6, 13]. Moreover, containers can even provide security benefits around the organization as well. Li et al [13] found that acquisition, development and management departments had a slight improvement in information security while another aspect such as access control and physical and environmental security has much greater improvement in information systems security.

## IV. SCALING AND RESOURCE MANAGEMENT

Resource management is important as the quality of resource management does not only affect the needs of the users but also directly affect the performance and the load of the system as well [9]. If the performance of the systems deteriorates, it may harm the business as well due to customers not being able to access the systems not responding in a timely manner. Resource management algorithms currently exist which can scale to either get more or fewer resources based on the demand.

Apart from the technical benefits by running microservices in containers that can scale, an organization can have monetary advantages as well by cutting down the cost of deployment as well. Guerrero et al [11] tested various algorithms and found out that with proper orchestration and scaling of microservices that the deployment cost of containers can be lowered and also increase the performance of the microservices running on the containers. Moreover, Villamizar et al [14] also found out that with the help of scalable microservice infrastructure, the cost of running microservices can be lowered. Running a monolith system in a scalable environment costs higher than the cost of running microservices in a scalable environment [14]. Moreover, with the help of serverless service providers, the cost of running microservices can be lowered even further [14].

The migration of containers is one of the ways that can manage the resources. Optimum minimum migration algorithm (OMNM) is one of the algorithms that aim to reduce the unnecessary migration of containers and ensures the load balancing of the clusters [9]. OMNM calculates the growth trends of each container and then determines which container is to be migrated [9].

Another way that resources are managed for containers is by resource provision from the physical host. The two-stage stochastic programming resource allocator (2SPRA) does this and it aims to provide resources to containers based on fluctuating incoming workloads which accommodates predefined SLO's on response latency [9]. As resources are limited and when the container requires more resources to function properly, pulling the image from an image registry will also require resources as well.

Even though many algorithms exist for resource provision, when allocating resources in a cloud environment there are multiple conflicting and influential objectives, therefore in the field, multi-objective optimization methods are widely used [9]. Such issues as time and computational resources being required to allocate more resources via a container also exist. Therefore, Algorithms that take considerations for resource allocation time have also been developed. Liu et al [3] proposed an algorithm that considers five factors, this was called Multi-objective container scheduling algorithm also named Multiport. The 5 factors which are considered are CPU usage of each node, memory usage of each node, the time which is spent when transmitting images over the network, relationship between containers and nodes and finally the clustering of containers [3]. Using the factors to calculate the most appropriate node to deploy the container was defined in Multiport via a matrix that is generated by scoring based on each of the five factors [3].

One of the biggest improvements in auto-scaling can be improved by only provisioning the required resources to each container so that containers are neither under-provisioned nor over-provisioned. The issue with either under-provisioning or over-provisioning was due to how auto-scaling was handled. In most cases, auto-scaling was done by using thresholds based on infrastructure metrics [12]. Taherizadeh et al [12] tackled this issue by developing an auto-scaling mechanism that uses dynamically changing thresholds based on both infrastructure and application metrics.

### A. *Serverless deployment*

Developments in virtualization technology have also paved the way for serverless deployments where server resources are allocated based on the demands [14]. Amazon Web Services (AWS) provides these services where the business only pays for the amount of resources that they use. Pérez et al [15] developed a framework that utilizes the serverless deployment which is called Serverless Container-aware ARchitecutre (SCAR) which runs on AWS. This feature by AWS is provided using containers where resources are allocated based on the requirements of the application running on it.

### B. *Server placement*

One of the most important factors for the performance when scaling or even when running microservices according to Sampaio et al [10] is the placement of each module in the microservice. According to Sampaio et al [10] the placement of microservices can either improve or degrade the performance as well as affect the utilization of resources. This is mainly due to communication affinities. Affinity is the relationship between two microservices, represented over time by the number and size of the messages that have been exchanged [10]. Affinities play a big role in the performance as microservices placed in two hosts can have high affinity leading to low performance due to high communication latencies. Moreover, resource utilization by microservices should also be considered when placing microservices in hosts as well, microservices which has high resource utilization can decrease the performance when placed in the same host [10]. By considering affinity and microservice resource utilization, a microservice placement mechanism called Runtime Microservice Placement (REMaP) was introduced by Sampaio et al [10]. REMaP was able to improve the performance and decrease resource utilization considerably in some of the tests which were conducted [10], which shows the importance of considering affinities and resource utilization when placing microservices. Even though Sampaio et al [10] studied how REMaP can handle the affinity issue within a microservice which is hosted on different servers, this can increase the cost as an organization will be paying for multiple servers. Multiple servers might be required by organizations as they grow but it is worth looking into better utilizing the resources with fewer resources. Guerrero et al [11] compared various scaling algorithms and found out that with proper orchestration and scaling the cost of deployment can be lowered. Guerrero et al [11] was able to achieve this by an algorithm which used an evolutionary algorithm called None-dominated sorting generic algorithm II (NSGA-II). REMaP and NSGA-II both have similarities as they both take communication affinity and resource utilization into consideration. On the other hand, unlike NSGA, REMaP takes resource utilization of other containers in the host server as well but the use of both the algorithms can also increase the cost compared to a serverless deployment as both require an SME to pay for the servers even if they are scaled down.

## V. LITERATURE REVIEW MATRIX

TABLE I. LITERATURE REVIEW MATRIX

| ***Article*** | ***Scope*** | ***Outcomes*** | ***Limitations*** |
|---|---|---|---|
| Performance comparison of container-based technologies for the Cloud | Compare and contrast currently existing container-based technologies that are used in the cloud environment | CPU cycles required for some utility tasks can cause decreases in performance even though, containers can almost give the same performance as the native machine | Core OS was not considered in the comparisons |
| Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications | An approach to optimize the deployment of microservices using containers | Compared to a greedy first fit algorithm the NSGA-II solution performed 309% more efficiently furthermore | Study of container orchestration optimization is limited as it is very new |
| Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures | Evaluate the infrastructure cost of using microservices | Shows that microservices can reduce the cost of infrastructure and the services designed to host scalable microservices cut the cost even more | Used only one serverless service provider (AWS lambda) |
| Dynamic Multi-level Auto-scaling Rules for Containerized Applications | Propose a new dynamic multi-level (DM) auto-scaling method with dynamic thresholds | In overall performed better than other auto-scaling methods | Unable to identify an effective monitor measurement interval |
| Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers | a framework for data centers to reduce the application deployment cost | The algorithm created was able to scale adaptively | - |
| Does migrating a monolithic system to microservices decrease the technical debt? | Monitor the technical debt of SME who converted monolith system to a microservice | Migration from a monolith to microservices helps reduce the technical debt. An initial spike in technical debt was seen when development started but afterward, technical debt grew slower than in a monolith system | Used a relatively small dataset (Only SonarQube) |
| SmartVM: an SLA-aware microservice deployment framework | Microservice deployment framework designed to streamline the development and deployment | Found advantages for using the newly developed framework, furthermore the newly developed | - |

| | | | |
|---|---|---|---|
| | of dynamically-scalable microservices | framework showed a 66% cost reduction compared to other microservice approaches | |
| Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud | Proposed a multi-objective container scheduling optimization model based on ant colony algorithm to solve the scheduling problem of microservice containers | Found advantages compared to other algorithms in reducing network transmission overhead and also when balancing load for clusters furthermore also found improvements in reliability | Only considers physical machine resource and load balancing is not considered |
| A new container scheduling algorithm based on multi-objective optimization | Proposes an algorithm which relies on 5 key factors to identify which node is best suited for container deployment | Compared to other algorithms observed a 7% increase in transactions per second and compared to other algorithms observed a 7.5% reduction in response time | - |
| The role of IT application orchestration capability in improving agility and performance | Investigate how IT application orchestration affect an organizations performance | Found out that IT application orchestration is vital to improving and organization IT portfolio | the focus was on a business unit level |
| Effects of virtualization on information security | Looks into post-virtualization security issues that are related to system security | Apart from Information systems acquisition, development and management others departments proved that the usage of virtualization provides great benefits for information security | Used a small sample size (13 experts) |
| Improving microservice-based applications with runtime placement adaptation | Look into the performance of microservices when placement is considered and come up with a mechanism (REMaP) to manage the placement of microservices automatically | REMaP was able to lower the utilization and increase the performance most of the time | Use predefined adaption triggers |
| Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective | Highlights different challenges and opportunities which comes with the adoption of microservices | Not all the organizations who adopt a microservice architecture will benefit from it equally | As a quantitative study was conducted, no particular focus of population was chosen |
| Application deployment using Microservice and Docker containers: Framework and optimization | Optimization of application deployment using microservices and docker containers | The proposed framework and the algorithm save more cost and provided more flexibility than the existing strategies | Containers have a reduced deployment cost and improved the user experience (better performance) |
| Serverless computing for container-based architectures | Introduction of a serverless container-aware architecture which can auto-scale for stateless jobs | A high throughput computing program model was developed | Only works on Amazon web services which had limited resources |

## VI. CONCLUSION

Microservices are proven to be efficient compared to monolith systems along with being much cheaper with proper scaling [14]. Even though this is the case an organization is required to invest in the skill sets of the development team to produce an efficient microservice [1]. If the investment is not made in the development team the produced microservice may even be less efficient then a monolith system. Moreover, the challenges of scaling are also an important decision an SME is required to make. Even though previously scaling was done mostly through VM's virtualization, containers are now becoming a more popular method [8]. The popularity is mostly due to containers being more lightweight. Due to the containers being more lightweight, they can scale more responsively [6, 9]. One of the reasons for this is because the containers can startup much faster than VM's due to containers being more lightweight [3, 4]. The scaling capabilities can help SME's to deliver online services without service degradation. Another benefit for the organization comes in the form of cost-saving. As VM's require more resources such as storage space as each VM requires an OS to be installed [4], this can slowly increase the cost of running a scalable microservice in a VM based solution. Furthermore, the scaling capabilities of Containers are more efficient than with VM's [4].s

The efficiency of scaling microservices can further be improved by using application-level metrics along with hardware matrices as well [6, 12]. Further decrease in cost was observed by scaling down when the resources are not required [9, 12].

Further reduction in cost was also discovered with the use of serverless deployment services such as AWS lambda [15].

AWS Lambda uses containers to scale based on the demands of the services that they run [15]. Leaving the microservices to be managed by a cloud service provider means that the SME loses some control over where each of the modules in a microservice is placed. The placement of the modules is important as it can have a positive or a negative impact on the performance as well [10, 11]. By considering the placement of the modules the performance can be increased [10, 11]. As the placement of the modules in the microservices increases the performance considerably, the downside is that it also can increase the cost compared to a serverless deployment solution. For an SME' the use of serverless deployment would have the most cost reduction while the use of scalable microservices hosted would allow them to scale based on the demand that they have while also being able to consider affinities when placing microservice modules.

## REFERENCES


[1] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges," J. Comput. Inf. Syst., 2018, doi: 10.1080/08874417.2018.1520056.

[2] V. Lenarduzzi, F. Lomio, N. Saarimäki, and D. Taibi, "Does migrating a monolithic system to microservices decrease the technical debt?," Journal of Systems and Software. 2020, doi: 10.1016/j.jss.2020.110710.

[3] B. Liu, P. Li, W. Lin, N. Shu, Y. Li, and V. Chang, "A new container scheduling algorithm based on multi-objective optimization," Soft Comput., 2018, doi: 10.1007/s00500-018-3403-7.

[4] X. Guan, X. Wan, B. Y. Choi, S. Song, and J. Zhu, "Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers," IEEE Commun. Lett., 2017, doi: 10.1109/LCOMM.2016.2644658.

[5] M. Queiroz, P. P. Tallon, R. Sharma, and T. Coltman, "The role of IT application orchestration capability in improving agility and performance," J. Strateg. Inf. Syst., 2018, doi: 10.1016/j.jsis.2017.10.002.

[6] T. Zheng et al., "SmartVM: a SLA-aware microservice deployment framework," World Wide Web, 2019, doi: 10.1007/s11280-018-0562-5.

[7] X. Wan, X. Guan, T. Wang, G. Bai, and B. Y. Choi, "Application deployment using Microservice and Docker containers: Framework and optimization," J. Netw. Comput. Appl., 2018, doi: 10.1016/j.jnca.2018.07.003.

[8] Z. Kozhirbayev and R. O. Sinnott, "A performance comparison of container-based technologies for the Cloud," Futur. Gener. Comput. Syst., 2017, doi: 10.1016/j.future.2016.08.025.

[9] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud," IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2924414.

[10] A. R. Sampaio, J. Rubin, I. Beschastnikh, and N. S. Rosa, "Improving microservice-based applications with runtime placement adaptation," J. Internet Serv. Appl., 2019, doi: 10.1186/s13174-019-0104-0.

[11] C. Guerrero, I. Lera, and C. Juiz, "Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications," J. Supercomput., 2018, doi: 10.1007/s11227-018-2345-2.

[12] S. Taherizadeh and V. Stankovski, "Dynamic multi-level auto-scaling rules for containerized applications," Comput. J., 2019, doi: 10.1093/comjnl/bxy043.

[13] S. H. Li, D. C. Yen, S. C. Chen, P. S. Chen, W. H. Lu, and C. C. Cho, "Effects of virtualization on information security," Comput. Stand. Interfaces, 2015, doi: 10.1016/j.csi.2015.03.001.

[14] M. Villamizar et al., "Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures," Serv. Oriented Comput. Appl., 2017, doi: 10.1007/s11761-017-0208-y.

[15] A. Pérez, G. Moltó, M. Caballer, and A. Calatrava, "Serverless computing for container-based architectures," Futur. Gener. Comput. Syst., 2018, doi: 10.1016/j.future.2018.01.022.