2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems

Feasibility of Serverless Cloud Services for Disaster Management Information Systems

Tayyaba Asghar
University of Lahore
University of Gujrat
Gujrat, Pakistan
tayyaba.asghar@uog.edu.pk

Saqib Rasool
Information Technology University
University of Gujrat
Gujrat, Pakistan
saqib@ieee.io

Muddesar Iqbal
School of Engineering
London South Bank University
London, UK
m.iqbal@lsbu.ac.uk

Zia ul Qayyum
University of Gujrat
Gujrat, Pakistan
Ziaqayyum@gmail.com

Adnan Noor Mian
Information Technology University
Lahore, Pakistan
adnan.noor@itu.edu.pk

George Ubakanma
London South Bank University
London, UK
ubakang@lsbu.ac.uk

***Abstract*—Serverless is the new generation of cloud services that supports the pay-per-use policy in true spirit by charging only for the execution time of the hosted code. Amazon introduced serverless service of Lambda in 2014 and it is consider as the most popular serverless cloud service till date. This paper focuses on the serverless cloud services of Lambda and elaborates the importance of Lambda based serverless cloud services for hosting the disaster management information systems (DMIS). We have identified two repeatedly occurring phases of the life cycle of a DMIS. These phases are high activity phase and low activity phase. Our findings state that serverless cloud services are well-suited for both of these phases of a DMIS. Serverless reduce the operational cost during the low activity phase by detaching the code from running containers and it improves the scalability during the high activity phase by quickly assigning the already available containers from the container pool. However, this all comes with the price of reduced QoS for initial requests and our experimental results reflect the extent of this quality degradation.**

***Keywords*-FaaS; Serverless; Lambda; QoS; DMIS; Disaster;**

I. Introduction

Serverless cloud services offer the fine-grained pay-per-use access to auto-scaling cloud resources and is achieved by presenting no charge policy for idle resources [1]. Serverless services are also known as FaaS (Function as a Service) [2] as each serverless service corresponds to a light-weight and stateless function. Each of these functions are inherently capable of quickly scaling against the bursty traffic and this scalability is entirely achieved by the cloud vendor without requiring any scalability knowledge by the user hosting that functions. Lambda [3] is the most popular FaaS and it was launched by Amazon in late 2014. Although Amazon deploys the serverless Lambda functions through containers but these are more robust and horizontally scalable [4] as compare to the Elastic BeanStalk, which is the server based container service by Amazon [4].

L. Silvestri [5] has identified the missing support of efficient resource allocation by cloud service providers for reducing the cost and ensuring the auto-scalability against two types of traffic viz. 1) bursty and 2) unpredictable. S. Hendrickson et al. [4] have proved that Lambda has solved the first problem of efficient auto-scaling against the bursty traffic by sharing a pool of containers among multiple instances of different applications. However, Lambda is not much effective for the unpredicted traffic.

J. Weinman [1] has identified the limitation of Lambda to handle the second problem as it is not capable of maintaining Quality of Service (QoS) against the unpredicted traffic. It is also suggested to evaluate the performance of Lambda against the personalized requirements, before using it in production. Reason for reduced QoS is cold start [6] which removes the idle functions and restores back these after the next request arrives. Although it ensures the optimal utilization of resources but it also reduces the QoS for the initial requests after a specific idle time period. More details of cold start are covered in section three.

Disaster Management Information System (DMIS) helps in the quick execution of the assistance operations just after the occurrence of a disaster. It also help in applying the lessons learned from previous disasters to improve the relief operations [7]. However, DMIS mostly remains idle during the time between two disasters and it suddenly generates the bursty traffic just after a disaster. We have termed both of these phases as low and high activity phases respectively. In this paper, we have presented that serverless can support both of these phases with a QoS tradoff for initial requests after an idle time period. Section two elaborates the evolution of virtualization which helps in understanding the serverless cloud services. Section three explains the effectiveness of serverless for low and high activity phases of DMIS. Section four enlists related work and is followed by conclusion.

978-1-5386-6614-2/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2018.00175

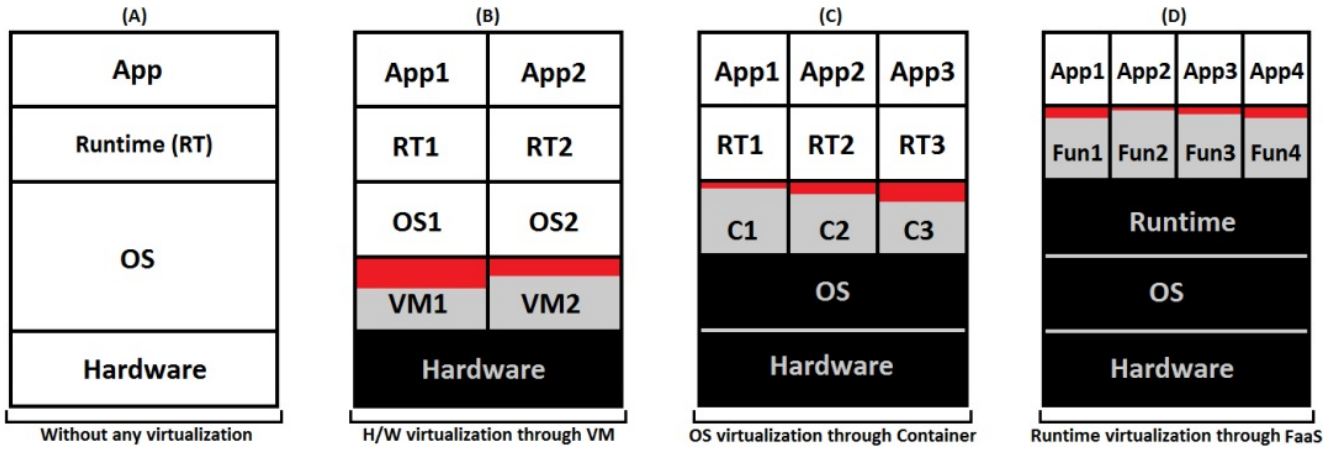

Figure 1. Evolution of virtualization from bare-metal server based hosting to virtualized serverless cloud services

II. Evolution of Virtualization

In early days, the computation was tightly bound to hardware and applications ran directly on hardware machines. It not only resulted in wastage of system resource but also became hurdle to horizontally scale the applications [8]. It also limited the applications to vertical scalability[9] which is achieved by increasing the hardware capacity of individual machines. In contrast, horizontal scalability can be achieved by distributing an application at different physical machines.

Attaining the horizontal scalability through physical hardware came up with twofold problem by 1) making it difficult to divide an application into distinct parts for deploying to multiple machines with minimum dependency between distributed codes and 2) by making it challenging to manage the sudden and unexpected hardware failures of physical machines. Virtualization provide solution to both of these problems by efficiently managing and distributing the physical resources and also by imposing different application development practices for supporting the horizontal and smooth scalability of deployed applications. Virtualization led to the beginning of cloud services.

Fig. 1 contains four sections where section A is showing a bare-metal server without any virtualization while other three sections represent three major virtualization phases in the history of cloud computing. There are four different colors used in Fig. 1 and each of these are explained below:
Background color of Black represents the virtualizaiton which is controlled by the cloud service provider.
Background color of Grey represents the platforms that are used by developers to host their codes on virtualized cloud infrastructure.
Background color of Red represents the wasted resources of the hosting platforms.
Background color of White represents the resources that are purely controlled by the developers that are hosting their applications.

A. Hardware virtualization through VM

VMs (Virtual Machines) offer the hardware virtualization which enables the sharing of a physical machine among multiple applications. However, it does not ensure the most efficient resource consumption. This is because each VM has its own OS (Operating System) and it holds the computational resources; even if the application is in idle phase. Moreover, if applications are in running phase, it is not very often that these will be utilizing all the designated resources of a VM. Red color in B part of Fig. 1 is representing the resources that are wasted due to the under-utilization of the designated resources.

B. OS virtualization through Container

Containers ensure the OS virtualization by sharing same kernel for offering multiple runtimes. Each runtime is further designated for an individual application. Containers are amongst the light-weight virtualization options with relatively less startup time as compare to the VMs. However, multiple container can share only a single type of OS-kernel [10]. Part C of Fig. 1 shows that more containers can be deployed with relatively less resources as compare to VM. Moreover resource wastage is also less in containers as compare to VMs and this is also clear from the difference of red color in part B and C of Fig. 1.

C. Runtime virtualization through FaaS (Serverless)

Both VM and containers do not impose application development patterns for achieving the horizontal scalability. However, runtime virtualization imposes the application developers to divide the application functionality into fine grained, light weight and stateless functions and deploy each function as an independent serverless service. Therefore, it is also known as Function-as-a-Service. It not only ensures the maximum sharing of resources but also maximize the consumption of allocated resources. It is also clear from the less amount of red color in section D of Fig. 1.

III. Feasibility of Serverless for DMIS

Life time of a DMIS can be divided into two distinct phases of 1) low activity phase during the time duration between two disasters and 2) high activity phase during the relief operation after a disaster. This section explains both of these two phases against the three discussed virtualization scenarios after explaining the QoS degradation due to cold start in serverless.

A. Reduced QoS due to cold start

QoS and SLA (Software Level Agreement) guarantees are considered among the major challenges of cloud services and these are more crucial for FaaS due to the ad-hoc assignment of containers to serverless functions. This autonomous process of resource management and assignment degrades the QoS as it is focused on reducing the operational cost of cloud vendors. A container pool is maintained by the Amazon and a Lambda function is assigned to one container upon the request. After the completion of function execution, container is released back to the container pool, if no other execution request is arrived for the same function.

Ad-hoc algorithm of resource management tries to assign the same container to the function on any future requests. However, there is no guarantee that same container is assigned again for the same request and it further increases the response time. This increased response time results in the QoS degradation because both QoS and execution time are inversely proportional.

We have performed extensive experiments with 1100 requests that are divided in to two sets. First set contains 500 time-varying requests with inter-transactional delay 1, 2, 3, 4 and 5 seconds each and second set contains 600 requests of inter-transactional delay of 10, 20, 30, 40, 50 and 60 each. Average of 100 requests against each of the 11 different inter-transactional delays is presented in Fig 2. It can be deducted from the presented results that the inter-transactional delay is the prominent factor for deciding the container detaching. For lesser inter-transactional delays, there were few chances of container detaching and it resulted in less average execution time. However, once a container is switched and function is assigned to a new container, it increases the execution time which resulted in QoS degradation.

Results presented in Fig. 2 were collected in June 2017 and were based on 1100 non-overlapping requests to a Lambda function. However, in case of overlapping requests, almost all initial request experience the similar degradation in QoS and this can further reduce the average QoS values presented in Fig. 2. Moreover, memory size of hosted function also effects the invocation time of a function. Detaching of code from the containers results in cold start and this problem can be solved by warming the serverless function by periodically invoking the deployed function.

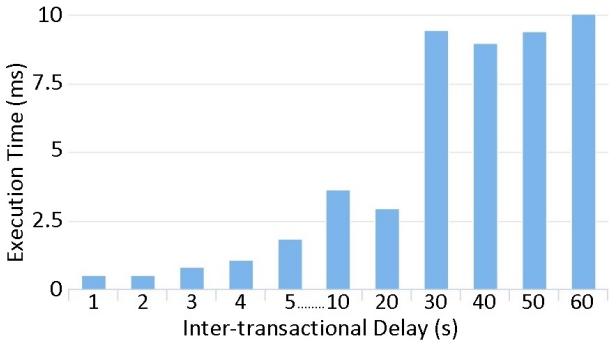

Figure 2. QoS degradation of a Lambda service against the time-varying workloads

B. Low activity phase

Information systems for disaster management generates bursty traffic during the disaster but remain almost idle for the time period between two disasters. Therefore, if a DMIS is hosted at VM or container based virtualized infrastructure then it will waste the resources during the idle time. However, in case of deployment on a serverless infrastructure, it will result in the reduction of operational cost due to cold start.

C. High activity phase

DMIS generates huge traffic during the relief operations after a disaster. If a DMIS is hosted on a VM or a container it will take time to launch new VM and containers and it reduces the overall QoS. This reduction in QoS will be experienced by the every newly launched instance due to the increased load on existing instances. Hence, QoS will not remain consistent throughout the relief operations. However, when same DMIS is hosted on serverless infrastructure, it will initially experience the degradation in QoS due to cold start. However, after the first few requests, QoS will remain consistent due to the scalable nature of the serverless cloud services. In case of serverless deployments, no time is wasted for launching new containers or VMs because a pool of containers is already maintained and just the allocation of code to a container needs to be done.

IV. Literature Review

In this paper, we have focused on finding the feasibility of serverless cloud services for hosting the DMIS. Serverless cloud services has already been used during the relief operations after a disaster. However, it is not exactly proposed for hosting DMIS and identifying its strengths against the low and high activity phases of a DMIS. Serverless cloud services are already used for real-time reporting from social media to support the decision making through flood mapping in Chennai [11]. Serverless has also been used for reducing the reunification time of children during the disaster scenarios [12].

V. Conclusion

Lambda is the most popular serverless cloud service by Amazon and this paper investigates the importance of Lambda for hosting the DMIS. We have identified two repeatedly occurring phases of a DMIS life cycle. First phase is the high activity phase which occurs during the relief operations of a disaster and second is the low activity phase which is based on the duration between two disasters. Serverless is good for efficient resource management by removing the running code of DMIS during the low activity phase for reducing the operational cost. It also gives more scalability during the high activity phase of DMIS by using an already maintained pool of containers. This paper has identified this strength serverless for supporting both low and high activity phases.

Acknowledgment

The authors are grateful for the support received from the transnational CiProVoT (Civil Protection Volunteers Training) project. The project is co-funded by ERAMUS+ programme of the European Union.

References

[1] A. Eivy, "Be wary of the economics of" serverless" cloud computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 6–12, 2017.

[2] J. Spillner, "Snafu: Function-as-a-service (faas) runtime design and implementation," *arXiv preprint arXiv:1703.07562*, 2017.

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

[4] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," *Elastic*, vol. 60, p. 80, 2016.

[5] V. Cardellini, E. Casalicchio, and L. Silvestri, "Service level provisioning for cloud-based applications," in *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications*. IGI Global, 2012, pp. 1479–1500.

[6] M. Stigler, "Understanding serverless computing," in *Beginning Serverless Computing*. Springer, 2018, pp. 1–14.

[7] J. Lee and T. Bui, "A template-based methodology for disaster management information systems," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. IEEE, 2000, pp. 7–pp.

[8] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*. Ieee, 2009, pp. 44–51.

[9] L. Mei, W. K. Chan, and T. Tse, "A tale of clouds: Paradigm comparisons and some thoughts on research issues," in *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*. Ieee, 2008, pp. 464–469.

[10] P. Andreetto, J. Astalos, M. Dobrucky, A. Giachetti, D. Rebatto, A. Rosato, V. Tran, M. Verlato, and L. Zangrando, "Egi federated platforms supporting accelerated computing."

[11] D. Sridhar and M. Priyaa, "Real-time flood mapping for disaster management decision support in chennai," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.

[12] K. Coleman, F. Esposito, and R. Charney, "Speeding up children reunification in disaster scenarios via serverless computing," in *Proceedings of the 2nd International Workshop on Serverless Computing*. ACM, 2017, pp. 5–5.