Performance Comparision between Scaling of Virtual Machines and Containers using Cassandra NoSQL Database

Sogand Shirinbab, Lars Lundberg, Emiliano Casalicchio
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
{Sogand.Shirinbab, Lars.Lundberg, Emiliano.Casalicchio}@bth.se

***Abstract*—Cloud computing promises customers the on-demand ability to scale in face of workload variations. There are different ways to accomplish scaling, one is vertical scaling and the other is horizontal scaling. The vertical scaling refers to buying more power (CPU, RAM), buying a more expensive and robust server, which is less challenging to implement but exponentially expensive. While, the horizontal scaling refers to adding more servers with less processor and RAM, which is usually cheaper overall and can scale very well. The majority of cloud providers prefer the horizontal scaling approach, and for them would be very important to know about the advantages and disadvantages of both technologies from the perspective of the application performance at scale. In this paper, we compare performance differences caused by scaling of the different virtualization technologies in terms of CPU utilization, latency, and the number of transactions per second. The workload is Apache Cassandra, which is a leading NoSQL distributed database for Big Data platforms. Our results show that running multiple instances of the Cassandra database concurrently, affected the performance of read and write operations differently; for both VMware and Docker, the maximum number of read operations was reduced when we ran several instances concurrently, whereas the maximum number of write operations increased when we ran instances concurrently.**

Keywords—Cassandra; Cloud computing; Docker container; Horizontal scaling; NoSQL database; Performance comparison; Virtualization; VMware virtual machine

I. INTRODUCTION

Today's modern data centers are increasingly virtualized where applications are hosted on one or more virtual servers that are then mapped onto physical servers in the data center. Virtualization provides a number of benefits such as flexible allocation of resources and scaling of applications. Scalability corresponds to the ability of a system uniformly to handle an increasing amount of work [1] [2] [3]**Error! Reference source not found.**. Nowadays, there are two types of server virtualization technologies that are common in data center environments, hardware-level virtualization and operating system level virtualization. Hardware-level virtualization involves embedding virtual machine software (known as hypervisor or Virtual Machine Monitor (VMM)) into the hardware component of a server. The hypervisor controls processor, memory, and other components by allowing several different operating systems to run on the same machine without the need for a source code. The operating system running on the machine will appear to have its own processor, memory, and other components. Virtual machines are extensively used in today's practice. However, during the last few years, much attention has been given to operating system level virtualization (also known as container-based virtualization or containerization). Operating system level virtualization refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances (also known as partitions or containers) instead of just one. As has been shown in Fig. 1, containers are more light weight than virtual machines, various applications in container share the same operating system kernel rather than launching multiple virtual machines with separate operating system instances. Therefore, container-based virtualization provides better scalability than the hypervisor-based virtualization [4].

Currently, two concepts are used to scale virtualized systems, vertical and horizontal scaling [5] [6] [7][8]. Vertical scaling corresponds to the improvement of the hardware on which application is running, for example addition of memory, processors, and disk space. While horizontal scaling corresponds to duplication of virtual servers to distribute the load of transactions. Horizontal scaling approach is almost always more desirable because of its advantages such as, no

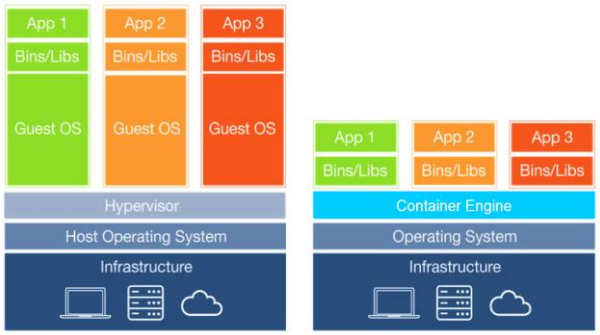


Fig. 1. Different of Virtual Machines and Containers Architecture

limit to hardware capacity, easy to upgrade, and easier to run fault-tolerance.

In our previous study, we explored the performance of a real application, Cassandra NoSQL database, on the different environments. Our goal was to understand the overhead introduced by virtual machines (specifically VMware) and containers (specifically Docker) relative to non-virtualized Linux [9]. In this study, our goal is to provide an up-to-date comparison of containers and virtual machine environments using recent software versions. In addition, explore how much horizontal scaling of virtual machines and containers will improve the performance in terms of the system CPU utilization, latency, and throughput. In this work, we have used multiple instances of the Cassandra running concurrently on the different environments.

The presented work is organized as follows: In Section II we discuss related work. Section III describes the experimental setup and test cases. Section IV presents the experimental results and we conclude our work in Section V.

II. Related Work

Both container-based and virtual machine-based virtualization technologies have been growing at a rapid space, and research work evaluating the performance aspects of these platforms provides an empirical basis for comparing their performance. Our previous research [9], has compared performance overheads of Docker containers, VMware virtual machines versus Non-virtualized. We have shown that, Docker had lower overhead compared to the VMware. In this paper, we try to expand our previous work and compare the two technologies; Container-based and Virtual Machine-based virtualization in terms of their scalabilities running Cassandra workload. There have not been many studies on both scalability and performance comparison between the two technologies. A comparison between Linux containers and AWS ec2 virtual machines is performed in [10]. According to their results, containers outperformed virtual machines in terms of both performance and scalability.

In [11]**Error! Reference source not found.**, the authors evaluated the performance differences caused by the different virtualization technologies in data center environments where multiple applications are running on the same servers (multi-tenancy). According to theirs study, containers may suffer from performance in multi-tenant scenarios, due to the lack of isolation. However, containers offer near bare-metal performance and low footprint. In addition, containers allow soft resource limits which can be useful in resource over-utilization scenarios. In [12], the authors studied performance implications on the NoSQL MongoDB during the horizontal scaling of virtual machines. According to their results, horizontal scaling affects the average response time of the application by 40%.

III. Evaluation

The goal of the experiment was that of comparing the performance scalability of the Cassandra while running it on multiple virtual machines versus on multiple containers concurrently.

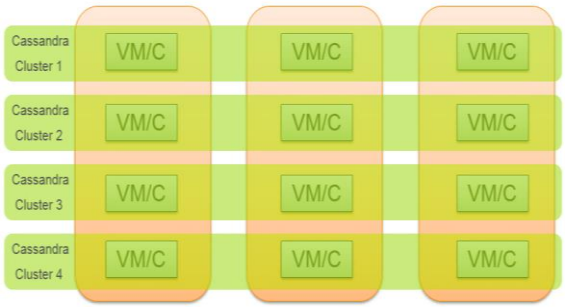


Fig. 2. Experimental Setup

A. *Experimental Setup*

All our tests were performed on three HP servers DL380 G7 with processors for a total of 16 cores (plus HyperThreading) and 64 GiB of RAM and disk of size 400 GB. Red Hat Enterprise Linux Server 7.3 (Maipo) (Kernel Linux 3.10.0-514.e17.x86_64) and Cassandra 3.11.0 are installed on all hosts as well as virtual machines. Same version of Cassandra used on the load generators. To test containers, Docker version 1.12.6 installed and in case of virtual machines VMware ESXi 6.0.0 installed. In total 4 times 3-node Cassandra clusters configured for this study (see Fig. *2*).

B. *Workload*

To generate workload, we used Cassandra-stress tool. The Cassandra-stress tool is a Java-based stress utility for basic benchmarking and load testing of a Cassandra cluster. Creating the best data model requires significant load testing and multiple iterations. The Cassandra-stress tool helps us in this endeavor by populating our cluster and supporting stress testing of arbitrary CQL tables and arbitrary queries on tables. The Cassandra package comes with a command-line stress tool (Cassandra-stress tool) to generate load on the cluster of servers, the cqlsh utility, a python-based command line client for executing Cassandra Query Language (CQL) commands and the nodetool utility for managing a cluster. These tools are used to stress the servers from the client and manage the data in the servers.

The Cassandra-stress tool creates a keyspace called keyspace1 and within that, tables named standard1 or counter1 in each of the nodes. These are automatically created the first time we run the stress test and are reused on subsequent runs unless we drop the keyspace using CQL. A write operation

inserts data into the database and is done prior to the load testing of the database. Later, after the data are inserted into the database, we run the mix workload, and then split up the mix workload and run the write only workload and the read only workload. In [9] [1], we described in detail each workload as well as the commands we used for generating the workloads, in this paper we have used the same approach for generating the workload.

C. Performance Metrics

The performance of Docker containers and VMware virtual machines are measured using the following metrics:

- CPU Utilization (percentage),
- Maximum Transactions Per Second (TPS), and
- Mean Latency (milisecond).

The CPU utilization is measured directly on the server nodes by means of sar command. The latency and maximum transactions per second (TPS) are measured on the client side, that are measured by the stress test tool. The term transactions per second refers to the number of database transactions performed per second.

D. Test Cases

1) One-Cassandra-three-node-cluster: In this case, one virtual machine/container deployed on each host running Cassandra application. All virtual machines/containers configured as one 3-node cluster.

2) Two-Cassandra-three-node-clusters: In this case, two containers/virtual machines deployed on each host running Cassandra application. Each container/virtual machine on each host belongs to its own 3-node cluster, so in total two 3-node clusters configured to run concurrently.

3) Four-Cassandra-three-node-clusters: In this case, four containers/virtual machines deployed on each host running Cassandra application. Each container/virtual machine on each host belongs to its own 3-node cluster, so in total four 3-node clusters configured to run concurrently.

In this experiment, we compare the performance of virtual machines and containers running different Cassandra workload scenarios, Mix, Read and Write. However, unlike our previous study [9], here we decided to set the replication-factor as three. In our test environment with three-node clusters, replication factor three means that each node should have a copy of the input data splits.

IV. Performance and Scalability Comparison

A. Transactions per second (tps)

Figure 3 shows transactions per second (tps) during write, read and mixed load. In this figure we summarized the total transactions per second from different number of Cassandra clusters running on Docker containers and VMware virtual machines. According to the results, overall in all cases Docker containers could handle higher number of database transactions per second than VMware virtual machines. In the case of the mixed load, Docker containers could handle around 25% more transactions per second than VMware virtual machines. In the case of only write load the difference is around 19% more for containers than virtual machines. While in the case of only read load, there is a huge difference of around 40% in the number of transactions per second between virtual machines and containers. Another aspect to consider according to the transactions per second results is that, running multiple instances of the Cassandra database concurrently, affected the performance of read and write operations differently; for both VMware and Docker, the maximum number of read operations was reduced when we ran several instances concurrently, whereas the maximum number of write operations increased when we ran instances concurrently. Note that increasing the number of Cassandra clusters did not have any significant impact on the number of transactions per second in the case of the mixed-load.

B. CPU utilization

Figure 4 shows the results of CPU utilization of multiple number of Cassandra clusters running on virtual machines and containers during write, read, and mix workloads. According to the results, in general CPU utilization of one cluster of virtual

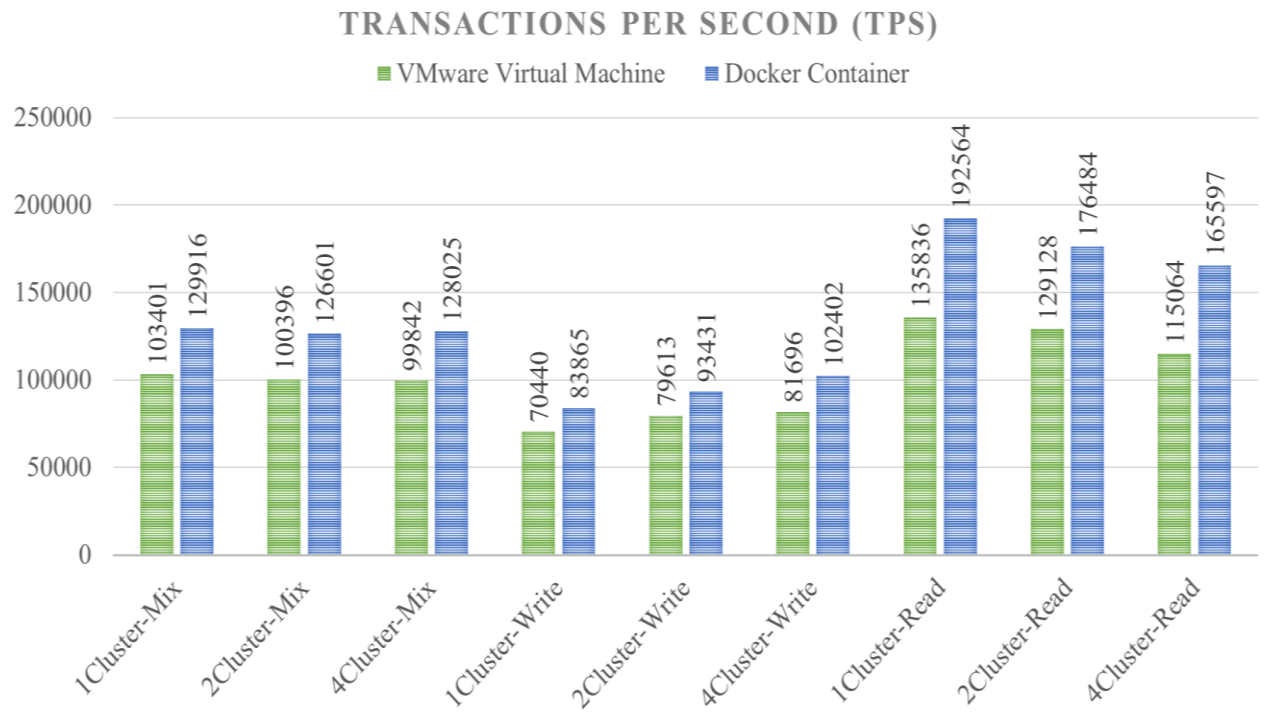
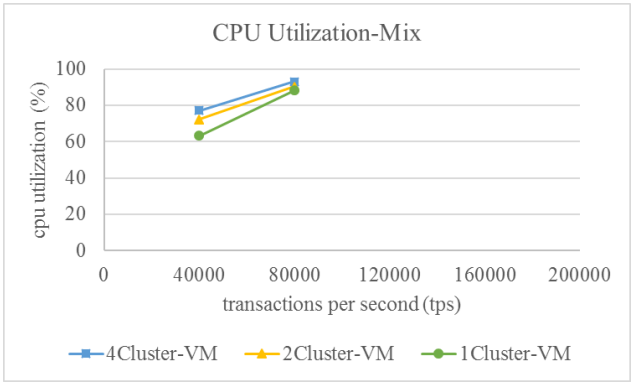
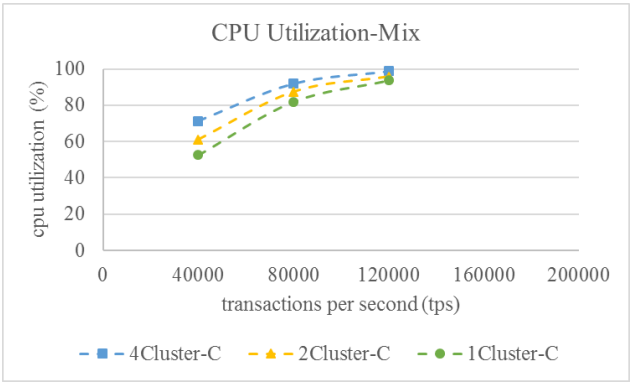


Fig. 3. Transactions per second (tps)

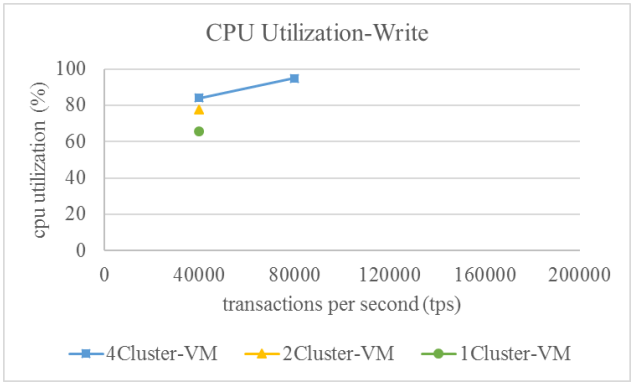
machines/containers are lower than two clusters and CPU utilization of two clusters is less than three clusters. It can be observed from the figures that, the overhead of running multiple clusters in terms of CPU utilization is around 10% for both containers and virtual machines. This overhead decrease as the load increases, one reason for this can be the background jobs that are running in Cassandra and as the load increases Cassandra by default delays these jobs since there are not enough resources available for executing the jobs. In addition, it can be observed from the figures that, the overall CPU utilization of containers is lower than virtual machines for all different workloads. Considering the mix workload CPU utilization of containers is around 15% lower than CPU utilization of virtual machines. The difference between CPU utilization of containers and virtual machines is around 12% for the write workload which is very close to the difference that we



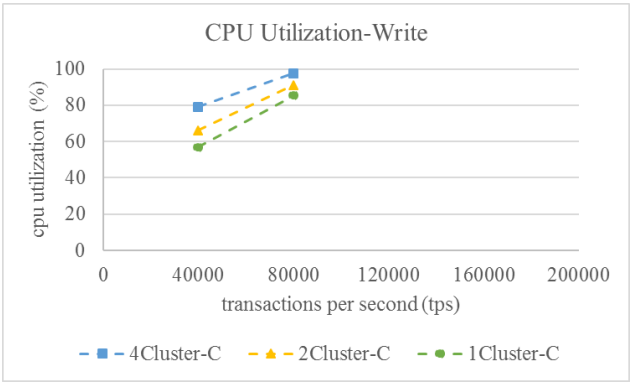
a.



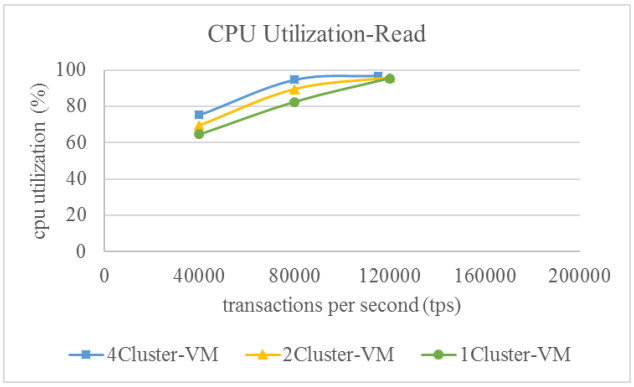
b.



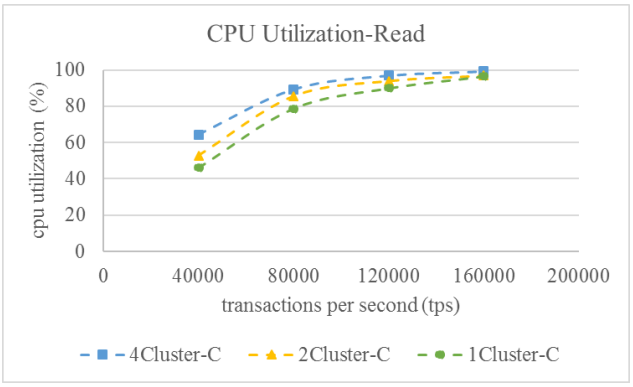
c.



d.



e.



f.

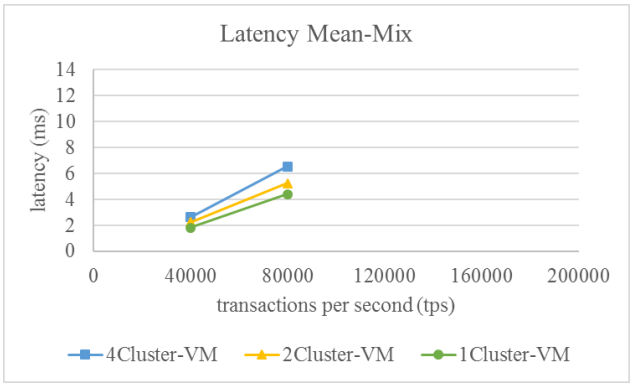
Fig. 4. CPU utilization results for Write, Read and Mix workload for multiple Cassandra clusters running on virtual machines and containers concurrently.

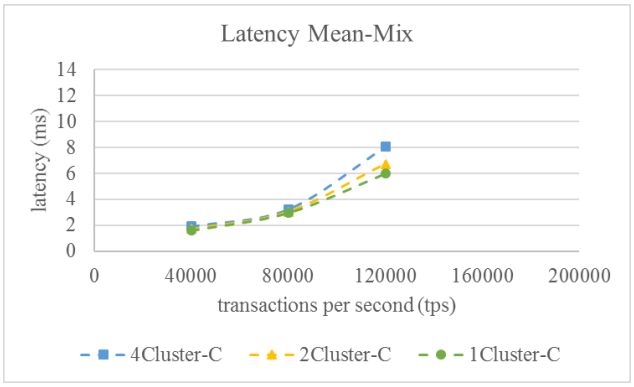
saw for the mix workload case. However, this difference is significantly higher for the read workload up to around 40%. According to these results, read operations utilize more CPU cycles on virtual machines than on containers.

C. *Latency*

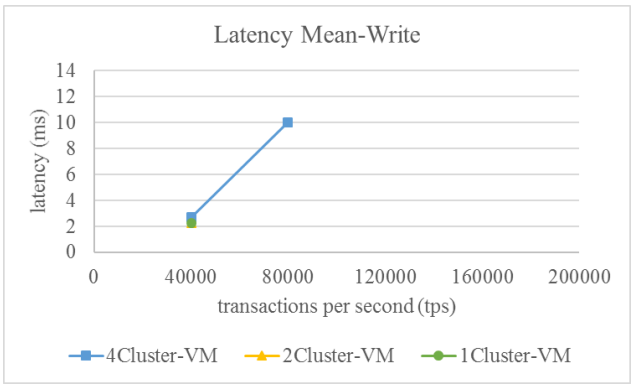
Figure 5 shows the results of latency mean of multiple number of Cassandra clusters running on virtual machines and containers during write, read, and mix workloads. As it can be observed from the figures, in general, the latency of containers is 50% lower than virtual machines as the load increases.

In the case of the mixed workload, the latency difference between having one cluster and two clusters is negligible. However, the latency difference between having one or two clusters compared with four clusters is around 33%. In the case of the write workload the difference between having containers.

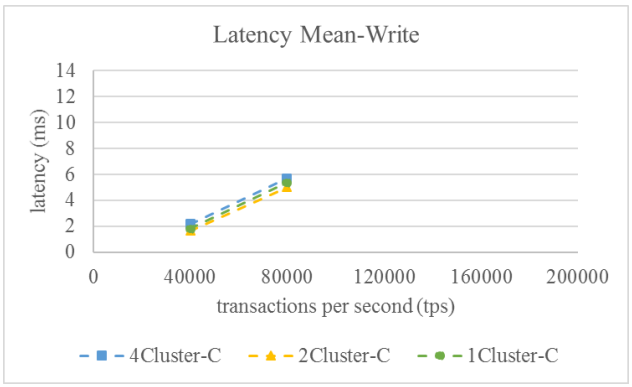




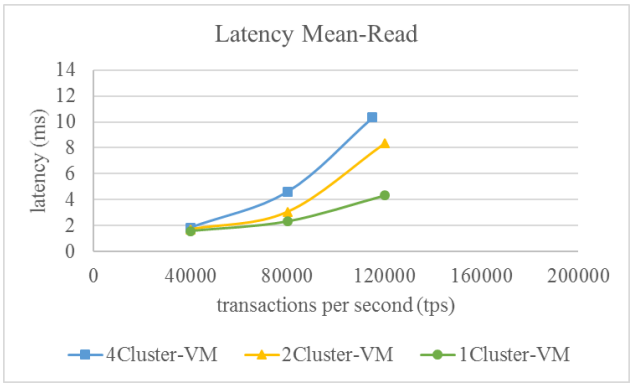
a.



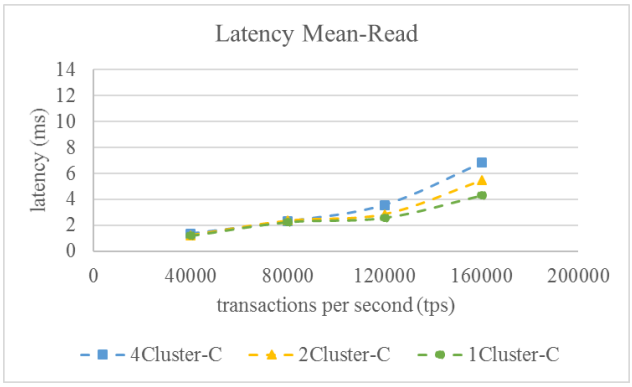
b.



c.



d.



e.

f.

Fig. 5. Latency mean results for Write, Read and Mix workload for multiple Cassandra clusters running on virtual machines and containers concurrently

However, for virtual machines, the latency becomes around 10ms in the case of 4 clusters when the tps is only 80k. Also, in the case of two clusters and 1cluster, since the cluster did not handle the load of 80k tps the latency is only shown for 40k tps which is around 2-3 ms. In the case of read workload, for the virtual machines the latency increases up to around 50% higher for the case with two clusters compared with one cluster. The latency increases up to around 20% for the case of four clusters compared with the case of two clusters and there is an increase of up to around 60% compared to the case of only one cluster. According to these results scaling would be very expensive for virtual machines in terms of latency mean which will have a negative impact on the application performance. However, in the case of containers the cost in terms of latency difference for having multiple clusters compared with one cluster is up to around 23%. According to the results, running multiple clusters inside containers will have less impact on the latency and the performance of the application (in this case Cassandra) than running multiple clusters inside virtual machines. The latency difference increases exponentially as the number of clusters increases as well as the load increases. The latency difference increases up to around 23% on containers and up to around 60% on virtual machines while having 100% read workload. The latency difference is negligible in the case of write workload. Also, there is a moderate latency difference in the case of mixed workload which is up to around 20% for virtual machines when the tps is 80k and up to around 25% for containers when the tps is 120k.

V. Discussions and Conclusions

In this study, we have compared the performance of running multiple clusters of the NoSQL Cassandra database inside Docker containers and VMware virtual machines. We have measured the performance in terms of CPU utilization, Latency mean and the maximum number of Transactions Per Second (TPS). According to our results, running Cassandra inside multiple clusters of VMware virtual machines was showing less performance in terms of maximum number of transactions per second compared to the Docker containers. The performance difference was around 20% lower during the mixed workload, around 16% lower during the write-only workload and around 29% lower during read-only workload. One reason for this could be that containers are lighter-weight compared to virtual machines, therefore there is a less overhead of the virtualization layer and this helps the application to get more resources and

performs better on containers than virtual machines. Another reason can be how a write and a read operation procedure works in Cassandra. In Cassandra, a write operation in general performs better than a read operation because it does not involve too much I/O. A write operation is completed when the data has been both written in the commit log (file) and in memory (memtable). However, a read operation may require more I/O for different reasons. A read operation first involves reading from a filter associated to sstable that might save I/O time saying that a data is surely not present in the associated sstable and then if filter returns a positive value, Cassandra starts seeking the sstable to look for data. In terms of CPU Utilization, the Cassandra application performs better on containers than on virtual machines. According to our results, the difference between CPU utilization on virtual machines is around 16% higher than containers during the mixed workload, around 8% higher during the write-only workload and around 32% higher during the read-only workload. In addition, the Cassandra application running inside virtual machines got up to around 50% higher latency than containers during the mixed workload. The difference became up to around 40% higher on virtual machines during the write-only workload compared to containers, also up to around 30% higher on virtual machines during the read-only workload compared to containers. As it has been discussed before in general the read-only workload is showing less performance than the write-only workload, and the impact of the different types of workloads on the performance in terms of CPU utilization is higher on virtual machines than containers.

However, considering the scalability aspects of the virtual machines and the containers, according to our results, containers scale better without loosing too much performance while virtual machines overhead is very high, and it has a negative impact on the performance of the application. This might differ depending on the application and the type of workload as we have seen during our experiments. Therefore, cloud providers need to investigate this issue while deploying both virtual machines and containers across data centers also at larger scale.

REFERENCES

[1] G. Huang et al., “Auto Scaling Virtual Machines for Web Applications with Queuing Theory,” in ICSAI conference, pp. 433-438, 2017.

[2] S. He et al., “Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning,” in AINA conference, pp. 15-22, 2012.

[3] A. Horiuchi, and K. Saisho, “Development of Scaling Mechanism for Distributed Web System,” in SNPD conference, pp. 1-6, 2015.

[4] F. Tseng et al., ”A Lightweight Auto-Scaling Mechanism for Fog Computing in Industrial Applications,” in IEEE Transactions on Industrial Informatics Journal, vol. PP, no. 99, pp. 1-1, 2018.

[5] W. Wenting, C. Haopeng, C. Xi, “An Availability-Aware virtual Machine Placement Approach for Dynamic Scaling of Cloud Applications,” in UIC/ATC conference, pp. 509-516, 2012.

[6] L. Chien-Yu, S. Meng-Ru, L. Yi-fang L. Yu-Chun, L. Kuan-Chou, ”Vertical/Horizontal Resource Scaling Mechanism for Federated Clouds,” in ICISA conference, pp.1-4 , 2014.

[7] S. Sotiriadis, N. Bessis, C. Amza, R. Buyya, “Vertical and Horizontal Elasticity for Dynamic Virtual Machine Reconfiguration,” in IEEE Transactions on Services Computing Journal, vol. PP, no. 99, pp. 1-14, 2016.

[8] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, P. Merie, ”Elasticity in Cloud Computing: State of the Art and Research Challenges,” in IEEE Transactions on Services Computing Journal, vol. PP, Issue. 99, pp 1-1, 2017.

[9] S. Shirinbab, L. Lundberg, E. Casalicchio, “Performance Evaluation of Container and Virtual Machine Running Cassandra Workload, ” in CloudTech conference, pp. 1-8, 2017.

[10] A.M. Joy, "Performance Comparison between Linux Containers and Virtual Machines," in ICACEA Conference, pp. 342-346, 2015.

[11] L. Chaufournier, P. Sharma, P. Shenoy, Y.C. Tay, "Containers and Virtual Machines at scale: A Comparative Study," in Middleware Conference, pp. 1-13, 2016.

[12] C. Huang et al., “The Improvement of Auto-Scaling Mechanism for distributed Database- A Case Study for MongoDB,” in APNOMS conference, pp. 1-3, 2013.