Published: 04 January 2019

ICDCN '19: International Conference on Distributed Computing and Networking
January 4 - 7, 2019
Bangalore, India

RESEARCH-ARTICLE

QoS and performance metrics for container-based virtualization in cloud environments

ISAM MASHHOUR JAWARNEH, University of Bologna, Bologna, BO, Italy

PAOLO BELLAVISTA, University of Bologna, Bologna, BO, Italy

LUCA FOSCHINI, University of Bologna, Bologna, BO, Italy

GIUSEPPE MARTUSCELLI, University of Bologna, Bologna, BO, Italy

REBECCA MONTANARI, University of Bologna, Bologna, BO, Italy

AMEDEO PALOPOLI, University of Bologna, Bologna, BO, Italy

View all

QoS and Performance Metrics for Container-based Virtualization in Cloud Environments

Isam Mashhour Al Jawarneh
DISI - University of Bologna
40136 - Bologna, Italy
isam.aljawarneh3@unibo.it

Paolo Bellavista
DISI - University of Bologna
40136 - Bologna, Italy
paolo.bellavista@unibo.it

Luca Foschini
DISI - University of Bologna
40136 - Bologna, Italy
luca.foschini@unibo.it

Giuseppe Martuscelli
DISI - University of Bologna
40136 - Bologna, Italy
giuseppe.martuscelli@unibo.it

Rebecca Montanari
DISI - University of Bologna
40136 - Bologna, Italy
rebecca.montanari@unibo.it

Amedeo Palopoli
DISI - University of Bologna
40136 - Bologna, Italy
amedeo.palopoli@unibo.it

Filippo Bosi
Imola Informatica
Via Selice, 66/A, 40026 Imola (BO), Italy
fbosi@imolinfo.it

ABSTRACT

Current cloud deployments heavily depend on hypervisor-based virtualizations. The overarching characteristics of Docker and containerization have given them a momentum in their widespread adoption recently as alternatives for their counterparts. However, little research has been done for comparing the QoS of both technologies, thus leaving the domain without widely accepted performance metrics. Aiming at informing the decision of the best fit in a specific cloud deployment, we have designed performance metrics that compare the performance of both designs in an in-house cluster deployed by using OpenStack. We focus on well-established representatives as baselines, including KVM from the hypervisor-based side, LXD from the container-based side in addition to Docker. Our results show that containerization is not a predominant fit-all solution that can always replace hypervisors for all cluster deployment and application scenarios. It can instead be thought of as a complementary solution to use for specific application scenarios that are constrained with conditions that are solved by containerization merits.

CCS CONCEPTS

Software and its engineering~Software performance• Software and its engineering~Functionality• Networks~Network performance modeling • Networks~Network performance analysis • Networks~Network measurement

KEYWORDS

Containerization, virtualization, cloud computing, hypervisor-based virtualization, QoS computing, OpenStack

ACM Reference format:

2019. QoS and Performance Metrics for Container-based Virtualization in Cloud Environments. In Proceedings of International Conference on Distributed Computing and Networking (ICDCN). ACM, New York, NY, USA, 10 pages. https://doi.org/0000

1 INTRODUCTION

Performance shortcomings of server-based architectures have motivated the emergence of cloud computing environments where several computing bare-metals collaborate by sharing their computing capacities, aiming at providing a transparent layer able to be utilized for big computing workloads and consequently at hiding the logistics of handling the underlying physical infrastructure. However, those advantages do not come for free as managing those commodities in a cohesive and robust manner requires costly consolidation.

To cope up with those challenges, virtualization technologies come into play, constituting solutions that abstract the underlying complexities of network system logistics and resource management mechanisms, thus allowing, e.g., to concurrently run multiple operating systems on the same server and to provide efficient resource utilization. There are two primary kinds of virtualization: i) the past-decade predominant Hypervisor-based virtualization systems induce performance overhead [1], that is caused by the need to load an entire operating system for each server instance; this has raised the demand for ii) a more lightweight virtualization such as containerization, which has rapidly evolved to become an adjunct to the hypervisor paradigm. The fanciest feature of containers is that they share the underlying kernel used by the operating system running the host machine while providing the same level of services as their predecessors. A cursory glance at those overarching characteristics of containerization promotes an intuitive facade that biases them over hypervisor counterparts [2].

However, at the current stage there is not enough quantitative analytics supporting this claim. Therefore, we have decided to deploy two resource-equal on-premise clusters with just-enough resources as a baseline to stress-test those two non-relational alternatives, hypervisor- and container-based, thus challenging their capacities by giving them an exposure to a great diversity of test loads. This way, we check how both behave in same resource-constrained environments in terms of processing, network and I/O. Our contribution is manifold. First, we define a set of metrics for measuring QoS and performance in cloud environments. Second, we provide guidelines that help practitioners decide what is more suitable and particularly when to use containerization instead of machine virtualization. Third, we show experimental results that ensure that the ingrained presumption that container-based virtualization performs better does not necessarily always hold.

The remainder of the paper is organized as follows. We first provide a background that summarizes traditional virtualization and the importance of containerization. We then define our performance metrics. In a later section, we employ our metrics using well-established benchmarks. Last two sections summarize recent related works, draw conclusions, and recommend future research directions.

2 BACKGROUND

2.1 Traditional Virtualization, Associated Challenges, and Requirements

Server virtualization is yet the de facto standard virtualization in cloud computing environments, where a virtualization layer allows multiple operating system instances to run concurrently on top of a single (or more) bare-metal server(s), dynamically partitioning and sharing available physical resources such as CPU, storage, memory and IO devices [3]. We here focus on Virtual Machines (VM [4]). Some work in the relevant literature refers to this as hypervisor-based virtualization of two possible primary types: i) bare-metal-based, where a hypervisor sits directly on top of underlying hardware and ii) a hosted version, where a hypervisor stands on top of a host operating system.

Server-based virtualization requires a full operating system to be loaded to each VM, which quickly become laborious, which motivates the emergence of containerization (or container-based virtualization). In simple terms, it involves encapsulating an application in a container within its own operating environment [5]. Containerization mechanisms induce a performance overhead negatively affecting QoS because of the cascading stratified architecture of its images [6]. Cloud resources are typically shared among multiple users, who are beating a path to those resources; thus, an efficient management is necessary. Therefore, performing a comparison between containers and VM regarding resources overload is essential.

2.2 Containerization

In simple terms, containerization is a virtualization method that aims at running applications in distributed computing environments [7]. Contrary to hypervisor-based virtualization, containerization abstract workloads from the underlying hardware, where multiple isolated systems (hereafter referred to as containers) run on top of a single kernel. Containers encapsulate only components that are necessary to run a specific software. A constellation of alternatives is available including Linux VServer, OpenVZ, and LXC [5]. Linux-VServer requires the host kernel to be patched, leaving host kernel and host computer as a single point of failure for all virtual servers. OpenVZ instead creates multiple isolated containers on a single physical server enabling better utilization and ensuring that applications do not conflict [9]. The successor for those two is Linux containers (LXC), which provide lightweight operating system virtualization [10].

LXD [11] has been patterned after LXC and gained an increased momentum recently, encapsulating its capabilities and aiming at delivering a multi-host container management with advanced features.

3 QOS-AWARE & PERFORMANCE METRICS FOR CLOUD CONTAINERIZATION

We have designed the following metrics for QoS-aware and performance evaluation of the suitability of containerization in the cloud.

3.1 CPU Analysis

Split into two benchmarks; CPU-power and CPU-contention. The first is concerned with the whole execution time of a single process that exploits the number of existing cores through the concept of multi-threading, whereas the CPU-contention test consists of analyzing the behavior of execution time by operating together multiple compute instances that compete to access the same shared computing resources.

3.2 Network Analysis

Network performance measures network service quality, including bandwidth, latency, and throughput. For the throughput and bandwidth analysis, we analyze the behavior by supporting the usage of both UDP and TCP communications, based on iperf. We split our network analysis tests test into three parts i) InterCloud, where a computing machine runs inside a cluster that consists of parallelly connected computing resources, whereas an Iperf server executes outside. ii) IntraCloud, where a computing machine runs inside the cluster and an Iperf Server executes in the same cluster but on a different physical server host. iii) IntraNode, where a computing machine runs inside a cluster and an Iperf Server is another computing machine that runs on the same physical server host.

3.3 Read/write Analysis

The aim of this test is to evaluate IO access patterns within each compute instance. We have chosen Bonnie++ benchmark, where we stress the IO capabilities by using heavy data loads that hit computing instances.

We have highlighted the behavior of the system under high data volumes by using application data sizes that are significantly larger than the system memory. So, for these experiments, the computing machines are set to 512MB of RAM and the files that they are working on have a size of 24GB (forty-eight times the size of the RAM).

3.4 Density Analysis

We define density in our analysis as the number of compute instances that a server can host without degrading its performance. The purpose of this benchmark is to investigate the behavior of CPU and memory usage by increasing the number of compute instances for every host.

3.5 Bootstrap Analysis

This analysis concerns the provisioning of a service request measuring the time needed to provide a complete service instance, booting it up and completing a snapshot instantiation.

4 EXPERIMENTAL RESULTS

4.1 Experimental Setup

We set up two different in-premise OpenStack clusters, employing two groups of similar servers for fair comparison. The physical servers constitute eight MicroServer HP Enterprise (HPE) Proliant Gen 8. They include 2x Intel (R) Celeron(R) CPU G16610T @ 2.30 GHz processors with 12 GB of RAM. In addition, we have deployed two cloud models: one using KVM as hypervisor and the other with LXD as a container.

4.2 Results

i) CPU Analysis

We employ four flavor sets, all containing 2048 MB RAM and 20 GB disk, while alternating the number of vCPUs between 1,2,4, and 6 for A, B, C, and D flavors respectively, the definition of flavors is about the two OpenStack clouds that we use to execute compute instances through KVM and LXD. To do the same for Docker, we exploit the functionality of Rancher to limit the resource capabilities of each Docker container. Fig. 1 illustrates the execution time against the number of threads when executing the same algorithm. This consists of compressing a file of 300 MB that is replaced in each compute instance. LXD outperform KVM by a tiny fraction. However, Docker performs unfavorably in this case. Fig. 2 elucidates the result of the CPU Contention Benchmark. Docker deployment outperforms LXD and KVM.

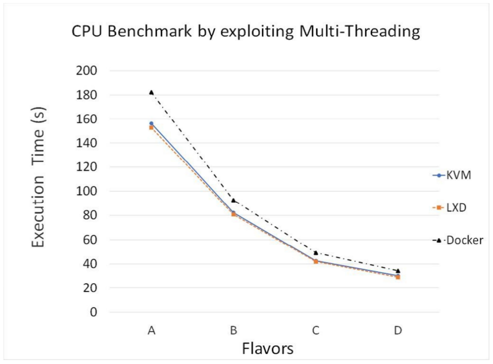

Figure 1: CPU Benchmark by exploiting Multi-Threading

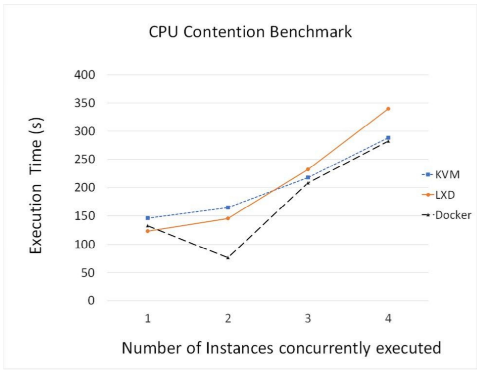

Figure 2: CPU Contention Benchmark

ii) Network Analysis

Fig. 3 (a and c) shows that Docker outperforms counterparts for intra-node scenarios in the language of TCP and UDP throughput. However, KVM has shown to have better performance in case of intracloud, and LXD in case of intercloud.

Fig. 4 depicts that server-based virtualization imposes a significant overhead in terms of latency (in case of intercloud, similar trends (despite not shown here because it's too speculative to clutter) also occurred in intracloud and intra-node) as LXD containerization and Docker outperforms KVM by orders of magnitude.

iii) Input/output

Fig. 5 elucidates that Docker is up to par from the perspective of performance achieved with IO benchmarks. The OpenStack KVM-based instance outperforms LXD by a tiny fraction for read workload and the opposite is true for write workloads.

iv) Density

The benchmark was structured utilizing the "stress" Linux utility that is simultaneously executed across each compute instance and the behavior of underlying physical resources has been monitored with Ganglia. Fig. 6a shows the behavior of CPU load against the number of compute instances that are scheduled on a single bare-metal OpenStack compute node.

Also, Fig. 6b illustrates the behavior of system memory when increasing the number of compute instances monotonically.

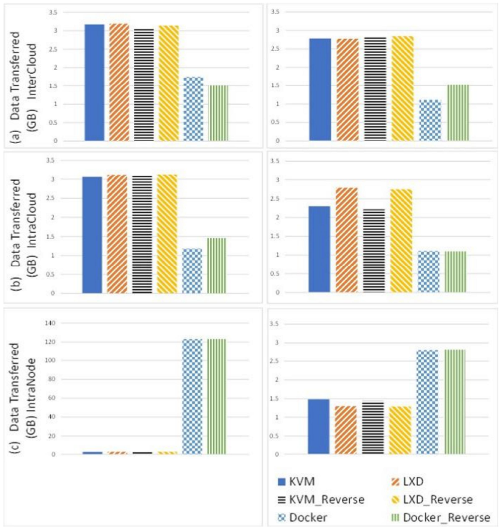

Figure 3: Throughput Analysis between TCP and UDP– Left column represents TCP, while right column elucidates UDP

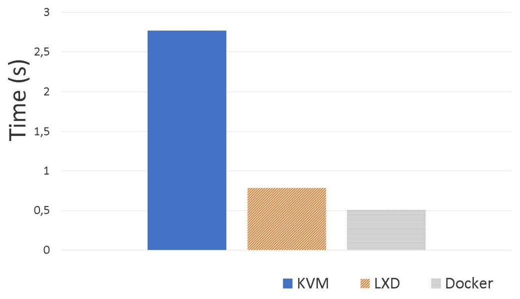

Figure 4: InterCloud Latency Benchmark

v) System

Container-based virtualization outperforms hypervisor counterparts in term of provisioning time.

This is mainly because VM require loading the full OS, which significantly counteracts the benefits of virtualization. Fig. 7 reports this comparison in detail.

4.3 Discussion

In short, the performance of Docker containerization alternates between 'the worst' over various performance indicators (such as CPU load and intercloud communication) and 'the best' for some other indicators (such as CPU contention).

Fig. 2 shows the spot where the performance degrades significantly for Docker, when increasing the number of instances from 1 to 2.

About network communication, containerization technologies outperform their counterparts for same cluster scenarios. However, those technologies perform less effectively when it comes to crossing the boundaries of their cluster communication, due to the need of complex communication's logistic handling for building interfaces that bridge gaps among various, often heterogeneous, cluster nodes.

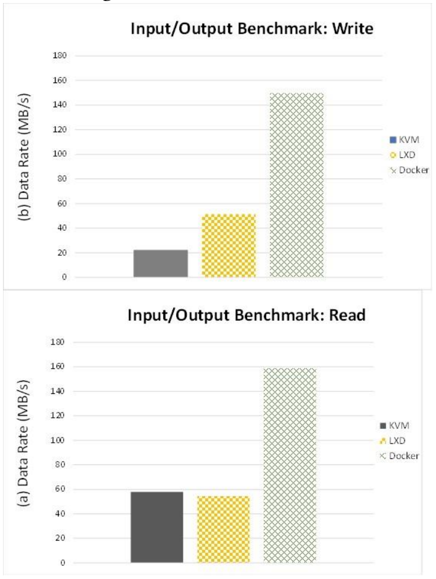

Figure 5: Input/output Benchmark

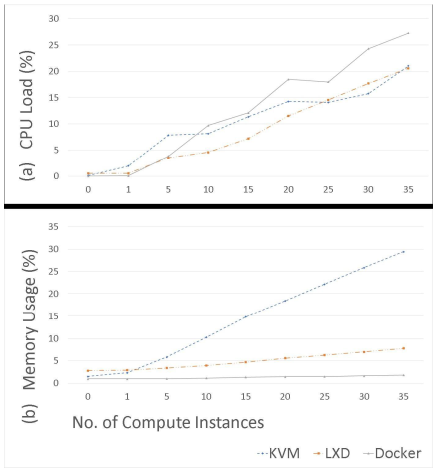

Figure 6: Instance Density Benchmark

This normally adds layers of complexity, specifically stemmed from hidden layers of translating bytecodes, higher to machine-level programming languages and protocols interdependencies. Figuratively speaking, Fig. 3a shows how Docker performed (almost) the worst, in both TCP and UDP workloads for InterCloud and IntraCloud scenarios, in the language of data transferred (measured in GB). Attentively speaking, this does not hold for all containerization systems, as LXD performs inversely for both deployments. Also, it performs inversely in IntraNode deployments. In all deployments, it has been noticed that, despite being a containerization system, LXD behave in a way that is similar to hypervisor-based counterparts.

On average, for IO tests, Docker outperforms KVM hypervisor server virtualization in addition to LXD containerization by at least 2x, which is quite significant. This shows that some containerization solutions (such as LXD) have to be considered unsuitable for cluster scenarios that have bursts in generating and consuming disk data in an unpredictable manner.

In short and as an overall consideration, despite some surprising results, our findings show that performance of virtualization technologies is highly dependent upon the mixture of configurations used for stress-testing, including primarily memory size, CPU power, and network configuration.

5 RELATED WORK

The work by [12] focuses on comparing container-based virtualization in two different settings, comparing two nodes interacting directly without a monitoring service and measuring the performance in the case of a presence of a management service layer.

Along the same lines, [6] applies a well-established set of benchmarks for confronting both virtualization paradigms. Within the same consortium, [13] conducts many experiments with various deployment based on cloudStack settings. However, they chiefly focus on network metrics. Harmonized with those, [14] evaluates the potential of migrating from VM to “Dockerization”. [15] has generated a set of synthetic benchmarks that aim at testing constrained-system behaviors when introducing containerizations as a virtualization technique.

6 CONCLUSIONS AND FUTURE WORK

We have analyzed the potentiality of containers as an alternative to VMs by proposing some original metrics for QoS and performance of containerization in the cloud. Docker is the principal player in this context, has taken a central position, and will continue this way at least for the foreseeable future. Our results demonstrate that containers achieve better performance as a general and on-average consideration. However, as they excel at some points, there are non-negligible technical elements that hinder containerization from becoming a fit-for-all alternative.

Convenient comparison between VMs and containerizations when it comes to hosting big data processing systems, such as Spark, is still missing. Comparison efforts are mostly ad hoc and patch efforts. There is a need for a systematic cohesive comparison that concludes the best option to opt for and embark on for complex big data processing workloads. We specifically mention the cases where an ecosystem such as Spark is deployed with containerizations and is used to process huge amount of burst and fast-arriving workloads in (near) real-time. Other avenues for investigation include our ongoing research work on the application point of view, aiming at evaluating the development process of enterprise applications by utilizing containerization.

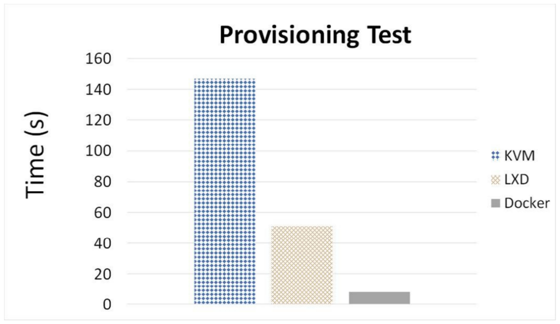

Figure 7: Instance Provision Benchmark

Acknowledgment

This research was supported by the SACHER (Smart Architecture for Cultural Heritage in Emilia Romagna) project funded by the POR-FESR 2014-20 (no. J32I16000120009) through CIRI.

REFERENCES

[1] N. Regola and J. C. Ducom, "Recommendations for Virtualization Technologies in High Performance Computing," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 409-416.
[2] D. Strauss, *the future cloud is container, not virtual machines* vol. 2013, 2013.
[3] X. Xu, H. Yu, and X. Pei, "A Novel Resource Scheduling Approach in Container Based Clouds," in *2014 IEEE 17th International Conference on Computational Science and Engineering*, 2014, pp. 257-264.
[4] V. N. Van, L. M. Chi, N. Q. Long, and D.-N. Le, "Performance Analysis of Network Virtualization in Cloud Computing Infrastructures on OpenStack," in *Innovations in Computer Science and Engineering*, Singapore, 2016, pp. 95-103.
[5] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, pp. 81-84, 2014.
[6] Z. Li, M. Kihl, Q. Lu, and J. A. Andersson, "Performance Overhead Comparison between Hypervisor and Container Based Virtualization," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, 2017, pp. 955-962.
[7] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs Containerization to Support PaaS," in *2014 IEEE International Conference on Cloud Engineering*, 2014, pp. 610-614.
[8] B. d. Ligneris, "Virtualization of Linux based computers: the Linux-VServer project," in *19th International Symposium on High Performance Computing Systems and Applications (HPCS'05)*, 2005, pp. 340-346.
[9] R. Rizki, A. Rakhmatsyah, and M. A. Nugroho, "Performance analysis of container-based hadoop cluster: OpenVZ and LXC," in *2016 4th International Conference on Information and Communication Technology (ICoICT)*, 2016, pp. 1-4.
[10] N. Memari, S. J. B. Hashim, and K. B. Samsudin, "Towards virtual honeynet based on LXC virtualization," in *2014 IEEE REGION 10 SYMPOSIUM*, 2014, pp. 496-501.
[11] J. S. Ma, H. Y. Kim, and Y. W. Kim, "The Virtualization and Performance Comparison with LXC-LXD in ARM64bit Server," in *2016 6th International Conference on IT Convergence and Security (ICITCS)*, 2016, pp. 1-4.
[12] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge," *IEEE Internet of Things Journal*, vol. 4, pp. 1019-1030, 2017.
[13] A. Vogel, D. Griebler, C. Schepke, and L. G. Fernandes, "An Intra-Cloud Networking Performance Evaluation on CloudStack Environment," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2017, pp. 468-472.
[14] N. Naik, "Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems," in *2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*, 2016, pp. 1-8.
[15] R. Morabito, "A performance evaluation of container technologies on Internet of Things devices," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2016, pp. 999-1000.