Quantifying the Isolation Characteristics in Container Environments

Chang Zhao[1], Yusen Wu[1], Zujie Ren[1], Weisong Shi[2], Yongjian Ren[1], and Jian Wan[3]

[1] Hangzhou Dianzi University, China
chaos_zhch@163.com, yusen.wu08@gmail.com, renzj@hdu.edu.cn, yongjian.ren@infocore.cn
[2] Wayne State University, USA
weisong@wayne.edu
[3] Zhejiang University of Science and Technology, China
wanjian@zust.edu.cn

Abstract. In recent years, container technologies have attracted intensive attention due to the features of light-weight and easy-portability. The performance isolation between containers is becoming a significant challenge, especially in terms of network throughput and disk I/O. Compared with virtual machines (VMs), containers suffer from a worse isolation as they share not only physical resources but also OS kernels. A couple of solutions have been proposed to enhance the performance isolation between containers. However, there is a lack of study on how to quantify the performance isolation between containers.
In traditional VM environments, the performance isolation is often calculated based on performance loss ratio. In container environments, the performance loss of well-behaved containers may be incurred not only by misbehaving containers but also by container orchestration and management. Therefore, the measurement models that only take performance loss into consideration will be not accurate enough. In this paper, we proposed a novel performance isolation measurement model that combines the performance loss and resource shrinkage of containers. We conducted a group of performance evaluation experiments based on an open-source container project Docker. Experimental results validate the effectiveness of our proposed model. Our results highlight the performance isolation between containers is different with the issue in VM environments.

Key words: containers, performance isolation, isolation measurement models

1 Introduction

Containers enable new ways to run applications by containerizing applications and services, making them portable, extensible, and easy to be transferred between private data centers and public clouds. Comparing with traditional virtual machines (VMs), containers have several advantages in terms of simplicity, lower-overhead, and light-weight. Docker, a user friendly open source application with high flexibility and efficiency, has been widely applied in the cloud industry and providing a compelling set of Platform-as-a-Service (PaaS) services, such as the Amazon, Alibaba, and so on.

However, containers suffer from a poor performance isolation as they share both OS kernels and physical servers. The misbehaving containers could easily overwhelm resource and interfere with the performance of the well-behaved containers. Therefore, the research on isolation improvement is becoming a hot topic in the field of container technologies [1] [2]. However, there is still a lack of study on how to quantify performance isolation, which is a preliminary step for isolation optimization.

A straightforward method to apply the performance isolation model for VMs is using the metric of performance loss ratio [3]. This kind of measurement models is based on an implicit assumption that the resource capacity of each VM remains constant. The pre-interfered performance and post-interfered performance statistics are collected from an identical VM without resource shrinkage. While in container environments, the life cycle and resources of containers are controlled by the container orchestrations. Therefore, the performance isolation measurement model in VM environments, is inapplicable for container environments.

In this paper, we proposed a comprehensive performance isolation measurement model that combines the performance loss and resource shrinkage of containers. The advantage of this model is that if the resource occupied by each container varies, the model can express the resource change, as well as the performance change. We conducted a group of performance evaluation experiments based on an open-source container project Docker. Experimental results validate the effectiveness of our proposed model. Our results highlight the performance isolation between containers is different with the issue in VM environments.

The rest of this paper is organized as follows. Section 2 gives a brief introduction about docker containers and isolation model. In Section 3, we describe the isolation model for containers. The validation experimental setups and results are presented in Section 4. Section 6 concludes this work and points out some future works.

2 Related Work

2.1 Containers technologies

In the mainstream cloud platforms, there are two different ways of virtualization: virtual machines and containers. Typical examples of virtual machines include KVM [4] and Xen [5]. Container is a new technology of operating system and application virtualization. It is widely used by cloud providers with the launch of Docker Project [6] and LXC [7]. A container is a lightweight, stand-alone, executable package of a piece of application. Containerized applications will always run the same, regardless of the environment, thereby promoting the portability of applications.

Containers running on an identical server share that the operating system kernel and server physical resources, and each container run as an isolated process in user space. Compared with VMs, containers start much quicker, and consume less CPU and memory spaces. Containers are an abstraction at the application layer that packages code and dependencies together.

Currently popular container technologies include Docker, OpenVZ [8], Linux-VServer [2] and so on [9]. Several cloud providers implemented container-as-a-service products

based on Dockers, and used Kubernetes [10] for container orchestration [11]. Container orchestration is the automated arrangement, coordination, and management of containers in their clusters. It controls different aspects of container life-cycles, such as placement and initial deployment, scaling and replication, and so on.

2.2 Isolation in virtualized environments

Performance isolation is one of the desirable features in virtualized environments. Well isolation contributes to guarantee enough resources for each tenant which is co-hosted with other tenants on an identical server [12] [13]. The resources include CPU cycles, memory space, and network bandwidth. To achieve a good isolation, a misbehaving tenant should be controlled to consume excessive resource and interfere other tenants.

In the field of traditional VMs, many research effort has been conducted to improve the performance isolation between VMs. Gupta *et al.* [14] developed a XEN-based monitoring system called XenMon. XenMon is designed to monitor CPU utilizations of each VM, and dynamically schedule the resource allocation of CPU shares to enhance isolation. Some solutions, such as Seawall[15] and SPIN[16], were proposed for enhancing the performance isolation in cloud data centers. Liu *et al.* [17] proposed an isolation measure model for VMs using multiple linear regression. This model combines the multiple performance indicators including resources utilization and number of VMs to calculate the isolation degree.

Eiras *et al.* [18] analyzed the performance of two open source virtualization solutions, KVM and Docker. The research results showed that the processing time for HTTP requests in Docker is lower than the ones of KVM. Morabito *et al.* [19] conducted a performance comparison of VMs and containers, to explore the strengths, weaknesses, and anomalies introduced by these different platforms in terms of processing, storage, memory, and network.

Shi *et al.* [20] proposed that in a distributed system the shared resource could be managed with different synchronization mechanisms such as a lock or token ring. However, in a smart EdgeOS, this performance isoltion might be more complicated. To solve this challenge, a well designed control access mechanism should be added to the service management layer in the EdgeOS.

In the field of container technologies, some researchers propose to improve the isolation by optimizing the schedule of jobs. For example, Xavier *et al.* [21] analyzed the performance interference suffered by disk-intensive workloads within noisy-perturbed containers. The authors in [21] proposed workload consolidation methods to reduce the performance interference between containers. Tang *et al.* [22] proposed a unified testing framework EIS to analyze on Docker, Lmctfy [23], ZeroVM [24] and as well as KVM. The authors in [22] compared these containers systems and conducted experiments to evaluate the virtualization technique alternatives for PaaS in clouds. Most studies focus on the impact of different load types on Docker isolation, or the degree of isolation of containers under some certain scenarios, high-performance computing [25, 26], workload-intensive environments and Hadoop framework [27]. Techniques and methods of performance isolation evaluation in container virtualization environment are not mature yet.

We noticed a few research works on performance isolation measured designed for traditional virtual machines [28]. These works often use the performance loss ratio to measure the isolation, which works on the assumption that the resource capacity of each virtualized machine is static. While for containers, the resource capacity for each container fluctuates. The resource is equally shared by containers. Therefore, the cause for performance losses for a container may not only due to the interference by the overloaded container, but also the decrease of resources. Therefore, the existing models for VMs are inapplicable in container environments.

In this work, we focus on how to measure and optimize the performance isolation of containers. We proposed a comprehensive performance isolation measurement model that combines the performance loss and resource shrinkage of containers. The advantage of this model is that if the resource occupied by each container varies, the model can express the resource changes, as well as the performance changes.

3 Performance Isolation Model

In this section, we will describe the measurement model for isolation characteristics between containers. Prior to the description of our proposed model, we would like to revisit the traditional isolation measurement model and clarify the motivation of this work.

3.1 Revisit the traditional isolation model

Traditional virtualized performance isolation models use the performance loss ratio [3][22][29], which is calculated as the formula 1. In the formula 1, P_i and P_j represent the pre-interfered performance and post-interfered performance of a specific VM being observed respectively. The resources of each VM resources are kept constant, so the performance loss is mainly caused by the performance isolation between VMs. While for containers, the resource capacity for each container fluctuates continuously. When a group of containers are running, their resources will be changed due to container orchestration and management.

$$I = \frac{|P_i - P_j|}{P_i} \tag{1}$$

In container environments, the performance loss of containers may be caused by the resources shrinkage. Therefore, we think that only using performance loss ratio as a metric of the isolation for containers is inapplicable for container environments. The performance isolation of the containers should take both performance loss and resource shrinkage into account.

Let us take an example. Suppose there are two servers with same capacity, S_1 and S_2. Both of them host two containers, S_1 hosts container A_1 and B_1, and S_2 hosts containers A_2 and B_2. The configurations of all the four containers are same. Both B_1 and B_2 are misbehaving containers. Suppose A_1 and A_2 suffer same performance loss ratio, such as twenty percent. Meanwhile, the resource consumed by A_1 decreases a considerable portion, and A_2 does not. The isolation of S_1 should be better than the one of the S_2, because A_1 consume less resources to achieve a specific performance.

3.2 A model combining performance loss and resource shrinkage

Our proposed model is designed to combine performance loss and resource shrinkage to measure the performance isolation between containers. As shown in the following model 2, the performance isolation is associated with P_{loss} and $R_{shrinkage}$. P_{loss} represents the performance loss degree of the containers, $R_{shrinkage}$ represents the resource shrinkage degree of the containers, and the larger value of P_{loss} and $R_{shrinkage}$ means the slower drop of the corresponding metrics. Through this model 2, we can normalize the performance isolation degree to the range of $[0, 1]$.

As shown in the model 2, given the same degree of resource shrinkage, if the performance loss is becoming large, the performance isolation is worse. But when the performance loss is the same, if the resource shrinkage is larger, the performance isolation is better. This is because when the performance degradation is same, the resources shrinkage is larger, indicating that the container resources fall more, the fewer resources are consumed. That means the containers can achieve the same performance while using fewer resources, which means the flexibility of containers resources is better.

$$I = f(P_{loss}, R_{shrinkage}) = \frac{(1+\beta^2) * P_{loss} * (1 - R_{shrinkage})}{\beta^2 P_{loss} + (1 - R_{shrinkage})} \tag{2}$$

The β in the formula 2 represents the scale factor used to adjust the weight of performance loss degree and resource shrinkage degree. A large β means that the performance loss is emphasized, while a small β means the resource elasticness is emphasized. When the β is ∞, it means that performance isolation is determined entirely by the degree of performance loss. And if the β is 0, performance isolation is determined entirely by the degree of resources shrinkage. By default, we can set β equal to 1.

The model 2 shows that the performance isolation of the containers is not only related to the degree of the performance loss of the containers, but also to the degree of resource shrinkage of the containers. For P_{loss} and $R_{shrinkage}$ in the model, we will use the formula 3 to calculate.

3.3 Quantifying performance loss

The performance of the containers is sensitive to the increased workload of other containers. Thus the container system composes primarily of two types of containers, affected containers and overloaded containers. If a container's workload is abnormally increasing and affecting the performance of other containers, we call this container an overloaded container.

Performance loss ratio does not take into account the curve progression. This implies that the performance is straight down and the parabola drops to the same value, resulting in the same result. But obviously when the performance is declining like the parabola, the performance isolation of the container system is better.

So in calculating P_{loss} in the model 2, we use the following method. Then we explain the isolation calculation method by Figure 1, the x-axis represents the performance of the overloaded containers, and the y-axis represents the performance of the affected containers.

In a well-isolated container system, the performance increasing of the overloaded containers does not affect the performance of other containers, as the green line shown in Figure 1. In a no isolation container system, as the performance of the overloaded containers increasing, the performance of the affected containers will drop linearly as the red line shown. And performance increasing of the overloaded containers is the same as the performance loss of the affected containers.Therefore, the slope of the red line in Figure 1 should be -1, and the values of P_{a_init} and P_{b_end} should be the same.

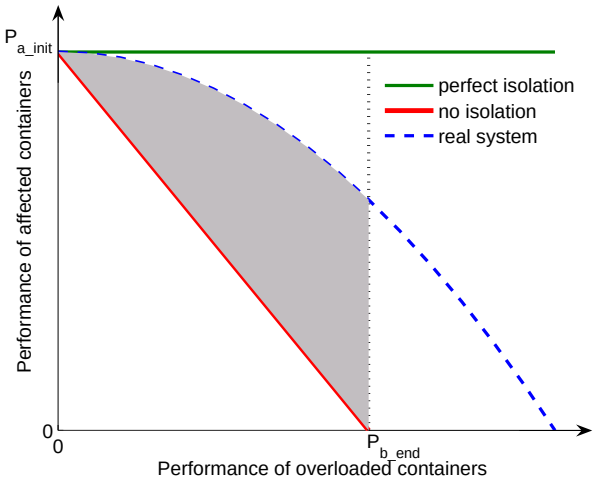

Fig. 1. Isolation curve including perfect isolation, no isolation and real system isolation.

However, in an actual container system, the performance of the affected containers is often not straight down, but as the blue dotted line in Figure 1, it shows a downward trend in the curve. Thus we found that the blue dotted line closer to the green line and the shadow area is larger, the performance loss degree is better, the blue dotted line closer to the red line and the shadow area is smaller, the performance loss degree is worse.We use the function $P_a = f(P_b)$ to represent the blue curve, then we define the calculation formula of the P_{loss} as the formula 3.

$$I = \frac{\int_0^{P_{b\_end}} f(P_b)dP_b - P_{b\_end} * P_{a\_init}/2}{P_{b\_end} * P_{a\_init}/2} \tag{3}$$

The resource shrinkage degree between the containers is similar to the performance loss degree, it is also affected by the overloaded containers. For the model 2, we can also use the formula refequ:fomula to calculate R_{loss}.

Due to the isolation of the containers is very weak, and the containers are sensitive to mutual influence, the containers' resources and performance change more frequently. This situation leads to instability in the experimental data, when calculating the P_{loss} and $R_{shrinkage}$. Since the calculated curve is not smooth, it fluctuates. So before we

calculate the results, we need to fit the measured experimental data to obtain a smooth curve. The abnormal value in the data should be processed to make the calculation result more accurate.

In this paper, we use the cubic spline interpolation method as a curve fit to obtain a smoothing curve to calculate P_{loss} and $R_{shrinkage}$. Spline interpolation is a kind of interpolation method commonly used and obtained a smooth curve. And the cubic spline interpolation is one of the most commonly used ones. Through the cubic spline interpolation, we can fit a smooth curve with the limited points to approximate the trend of the metrics.

4 Validation of the Model

In this section, we will describe the experiments to validate the measurement model. And we will experiment and evaluate the performance isolation from the workload of the affected containers and the total number of containers.

4.1 Experiment Setups

The performance of the experimental record is the common metric TPS (the number of transactions per second) in the database performance test, and we use CPU occupancy rate as a resource metric. The database is selected as MySQL 5.1, and the workload is generated by Sysbench.

Sysbench is a modular, cross-platform, multi-threaded benchmarking tool that is used to evaluate the database workload under a variety of system parameters and to evaluate database performance by random reading and writing to the database [30].

The operating system in the experiment is Centos7, and Docker version is 1.12.6. Docker is configured without any quotas on the use of the resources, that means a container can use as many resources are available.

4.2 Methodologies

The containers for each experiment consist of two parts, the affected containers and the overloaded containers, and the number of containers in both parts is equal. The performance and resource of the two kinds of containers take the average separately.

In the experiment, the affected containers run a stable database workload task, and the database workload task of the overloaded containers is gradually increased, and the performance metric and resource metric are recorded respectively.

For the obtained multiple sets of data, we fit the data to derive the curve by using the cubic spline interpolation method, and calculate the P_{loss} and $R_{shrinkage}$ by the formula 3, finally we use the isolation measure formula 2 to calculate the final performance isolation.

4.3 Validation of Isolation Changes

In theory, when the total number of containers is different, the degree of impact between containers is also unlike. When the number of containers is small, the influence is small between the containers, then the performance isolation is good. And when the number of containers increased, the competition between the container increased, and the containers interact with each other frequently, so the performance isolation is relatively poor.In order to measure the effect of the number of containers on system performance isolation, we conducted the following experiments.

We will perform 4 groups of experiments respectively in the same system, and the total number of containers was different between these experiments.We calculate the performance isolation, when the total number of containers is 2, 4, 8, and 16 respectively.For each case, we conducted eight experiments. Figure 2 shows the results of the case 1 when the total number of containers is 8.

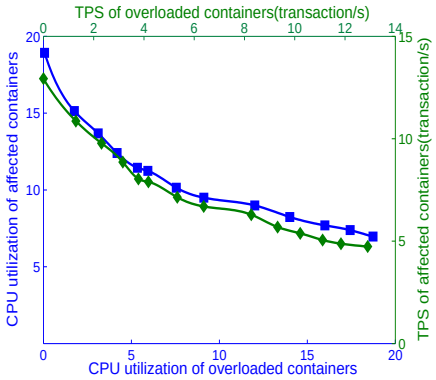

Fig. 2. Isolation curve of resource and performance with 8 containers in case1.

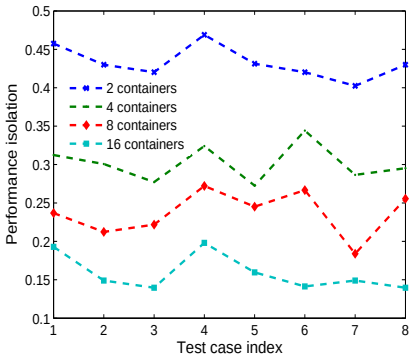

Fig. 3. Performance isolation with different number of containers.

As Figure 2 shows, when the total number of containers is 8, the container CPU resources and TPS performance decline curve.We can see from the figures, with the workload of the overloaded containers increasing, the affected containers decline sharply at the beginning, then gradually slow down. Using the curves in Figure 2, we can calculate P_{loss} and $R_{shrinkage}$ when the total number of containers is 8. And then through the isolation model to calculate the performance isolation. Similarly, the results of the other three groups of experiments are computed by this way.

Experimental results are shown in Figure 3, we can find that when the total number of containers is the same, performance isolation of the container system is relatively stable. While the total number of containers is different, the performance isolation of the system showed a big difference. And performance isolation of the containers decreases with the increase of the number of containers. The experimental results are in agreement with our suppose and verify the accuracy and validity of the performance isolation measure model.

4.4 Model Comparison

There are two types of containers in the container system, affected containers and overloaded containers. And the workload of the affected containers will have an impact on the performance isolation in a certain degree. When the workload on the affected containers is low, the overloaded containers can easily grab the resources from the affected containers and have a significant impact on the affected containers. In this case, the affected degree of the containers becomes larger and the performance isolation is poor. And if the workload of affected containers is high, then the overloaded containers are not easy to interfere with the affected containers. This time the performance isolation is relatively good. The purpose of these experiments is to measure the impact of the affected container workload on performance isolation.

The experiment controls the workload of the containers by controlling the number of threads of Sysbench. When the number of threads increases, the workload on the containers increases. The number of threads in the experiment was 4, 6, 8 and 10, respectively. The number of affected containers and overloaded containers for each experiment is 4.

In order to compare with the traditional isolation model of the performance loss ratio, we will use the performance loss ratio and our measurement model to calculate the performance isolation, and then the experimental results were compared. In each case, eight experiments were performed and 64 sets of data were recorded.

Experimental results are shown in Figure 4. Model1 represents our model, and model2 represents the traditional performance loss ratio. Mode1-4 means the result of 4 threads in our model. The top four lines are the result of the decline of traditional performance loss ratio, and the bottom four lines are the result of our model. As we can see from Figure 4, when the affected containers workload is different, the difference of the performance loss ratio is small, so it can not effectively measure the changes of the performance isolation. And our model presents a large difference, the difference of the performance isolation corresponding to the different workload is more obvious.

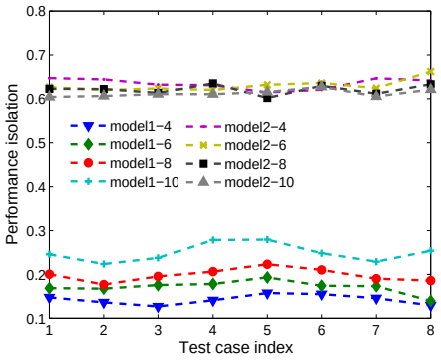

Fig. 4. Performance isolation with different workload of affected containers.

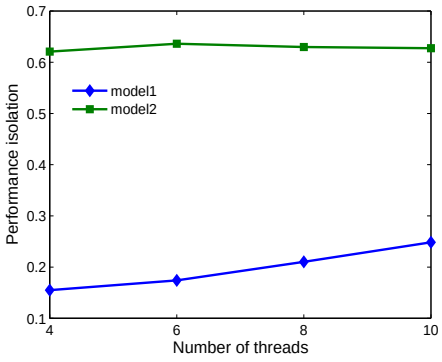

Fig. 5. Performance isolation with different workload of affected containers in case 6.

We will take out the eight sets of data of the case 6 in Figure 4, and drawn into Figure 5. So Figure 5 shows the results of the performance loss ratio and our metric model in the first experiment. From Figure 5, we found that both methods show an upward trend in performance isolation, but the results of the performance loss ratio are less obvious, and our model clearly shows that the performance isolation increases as the workload increases in the affected containers.

We validate our model through experiments and compare it with the model of performance loss ratio. The experimental results show that in some cases, the result of our model is more likely to reflect the performance isolation trend of the containers than the performance loss ratio in container system. The result shows that in the container environment, only considering the performance loss is not an accurate measurement of performance isolation, and we need to join the measurement of resources.

5 Discussion

5.1 Performance isolation vs. resource elasticity

Resource elasticity is the feature of dynamic, flexible and frequent resizing of resources that are provided to an application by the virtualized platform. High elasticity can be considered as a key benefit of the container-based virtualization. We should notice that resource elasticity and performance isolation may have conflicts when the server is overloaded because of one misbehaving container. In other words, there is a tradeoff between performance isolation and resource elasticity. Therefore, when a method for optimizing the isolation is designed, it is quite necessary to guarantee an appropriate elasticity. That is why we combine the resource changes into the measurement model proposed in this work.

5.2 Resource management optimization

A simple way to optimize the performance isolation is to implement an effective resource management for containers. The central problem for resource management is allocating resources to support the varying demand for these containers. One approach is to maintain a pool of idle resource capacity that is allocated to containers as they suffer overload. The problem of dynamic resource resources is to do it in time according to the variant of workload. Using the model proposed in this work, system designers can evaluate not only performance isolation, but also the effectiveness of resource management.

5.3 Workload-aware container orchestration

Container orchestration is container scheduling, cluster management, and provisioning of more hosts in containers environment. To achieve effective orchestration and reduce the performance interference between containers, the first step is to characterize workload within containers. Their workloads may belong to one of the categories, including CPU-intensive, IO-intensive and so on. For each container, its workload can be modeled as a vector of resource usage. The workload model of containers will be used to guide container consolidation and mitigate the performance interference.

6 Conclusions

In this paper, we proposed a performance isolation model for container-based virtualization systems. The performance isolation model combines the performance loss ratio and resource shrinkage. We implemented a continuous dynamic model to calculate the changes of resources and performance, to improve the accuracy and effectiveness of the results. Finally, we conducted a group of experiments to validate our proposed model.

7 Acknowledgement

This research is supported by the National Natural Science Foundation of China (No.61572163). Weisong Shi is in part supported by National Science Foundation (NSF) grant CNS-1563728.

References

1. Ma, S., Jiang, J., Li, B., Li, B.: Maximizing container-based network isolation in parallel computing clusters. In: Network Protocols (ICNP), 2016 IEEE 24th International Conference on, IEEE (2016) 1–10
2. Xavier, M.G., Neves, M.V., De Rose, C.A.F.: A performance comparison of container-based virtualization systems for mapreduce clusters. In: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on, IEEE (2014) 299–306
3. Huber, N., von Quast, M., Hauck, M., Kounev, S.: Evaluating and modeling virtualization performance overhead for cloud environments. In: CLOSER. (2011) 563–573
4. Zhang, B., Wang, X., Lai, R., Yang, L., Wang, Z., Luo, Y., Li, X.: Evaluating and optimizing i/o virtualization in kernel-based virtual machine (kvm). In: IFIP International Conference on Network and Parallel Computing, Springer (2010) 220–231
5. Masood, A., Sharif, M., Yasmin, M., Raza, M.: Virtualization tools and techniques: Survey. Nepal Journal of Science and Technology **15**(2) (2015) 141–150
6. Bernstein, D.: Containers and cloud: From lxc to docker to kubernetes. IEEE Cloud Computing **1**(3) (2014) 81–84
7. Linux Container-LXC: (2017)
8. Babu, A., Hareesh, M., Martin, J.P., Cherian, S., Sastri, Y.: System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver. In: Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on, IEEE (2014) 247–250
9. Kozhirbayev, Z., Sinnott, R.O.: A performance comparison of container-based technologies for the cloud. Future Generation Computer Systems **68** (2017) 175–182
10. Medel, V., Rana, O., Arronategui, U., et al.: Modelling performance & resource management in kubernetes. In: Proceedings of the 9th International Conference on Utility and Cloud Computing, ACM (2016) 257–262
11. Netto, H.V., Lung, L.C., Correia, M., Luiz, A.F., de Souza, L.M.S.: State machine replication in containers managed by kubernetes. Journal of Systems Architecture **73** (2017) 53–59
12. Krebs, R.: Performance Isolation in Multi-Tenant Applications. PhD thesis, Karlsruhe Institute of Technology (2015)
13. Krebs, R., Momm, C., Kounev, S.: Metrics and techniques for quantifying performance isolation in cloud environments. Science of Computer Programming **90** (2014) 116–134

14. Gupta, D., Cherkasova, L., Gardner, R., Vahdat, A.: Enforcing performance isolation across virtual machines in xen. In: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer (2006) 342–362
15. Shieh, A., Kandula, S., Greenberg, A.G., Kim, C.: Seawall: Performance isolation for cloud datacenter networks. In: HotCloud. (2010)
16. Li, X.H., Liu, T.C., Li, Y., Chen, Y.: Spin: Service performance isolation infrastructure in multi-tenancy environment. In: International Conference on Service-Oriented Computing, Springer (2008) 649–663
17. Liu, W., Feng, G.: Study of quantifying performance isolation of virtualization system. Computer Engineering & Applications (2015)
18. Eiras, R.S., Couto, R.S., Rubinstein, M.G.: Performance evaluation of a virtualized http proxy in kvm and docker. In: Network of the Future (NOF), 2016 7th International Conference on the, IEEE (2016) 1–5
19. Morabito, R., Kjällman, J., Komu, M.: Hypervisors vs. lightweight virtualization: a performance comparison. In: Cloud Engineering (IC2E), 2015 IEEE International Conference on, IEEE (2015) 386–393
20. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet of Things Journal **3**(5) (2016) 637–646
21. Xavier, M.G., De Oliveira, I.C., Rossi, F.D., Dos Passos, R.D., Matteussi, K.J., De Rose, C.A.: A performance isolation analysis of disk-intensive workloads on container-based clouds. In: Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on, IEEE (2015) 253–260
22. Tang, X., Zhang, Z., Wang, M., Wang, Y., Feng, Q., Han, J.: Performance evaluation of light-weighted virtualization for paas in clouds. In: International Conference on Algorithms and Architectures for Parallel Processing, Springer (2014) 415–428
23. Dua, R., Raja, A.R., Kakadia, D.: Virtualization vs containerization to support paas. In: Cloud Engineering (IC2E), 2014 IEEE International Conference on, IEEE (2014) 610–614
24. Rad, P., Lindberg, V., Prevost, J., Zhang, W., Jamshidi, M.: Zerovm: secure distributed processing for big data analytics. In: World Automation Congress (WAC), 2014, IEEE (2014) 1–6
25. Xavier, M.G., Neves, M.V., Rossi, F.D., Ferreto, T.C., Lange, T., De Rose, C.A.: Performance evaluation of container-based virtualization for high performance computing environments. In: Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, IEEE (2013) 233–240
26. Chung, M.T., Quang-Hung, N., Nguyen, M.T., Thoai, N.: Using docker in high performance computing applications. In: Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on, IEEE (2016) 52–57
27. Rizki, R., Rakhmatsyah, A., Nugroho, M.A.: Performance analysis of container-based hadoop cluster: Openvz and lxc. In: Information and Communication Technology (ICoICT), 2016 4th International Conference on, IEEE (2016) 1–4
28. Walraven, S., Monheim, T., Truyen, E., Joosen, W.: Towards performance isolation in multi-tenant saas applications. In: Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing, ACM (2012) 6
29. Matthews, J.N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., McCabe, M., Owens, J.: Quantifying the performance isolation properties of virtualization systems. In: Proceedings of the 2007 workshop on Experimental computer science, ACM (2007) 6
30. Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An updated performance comparison of virtual machines and linux containers. In: Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on, IEEE (2015) 171–172