An Introduction to Docker and Analysis of its Performance

Article · March 2017

CITATIONS
46

READS
10,866

3 authors:

Babak Bashari Rad
Asia Pacific University of Technology and Innovation
49 PUBLICATIONS **384** CITATIONS
SEE PROFILE

Harrison John Bhatti
Halmstad University
6 PUBLICATIONS **60** CITATIONS
SEE PROFILE

Mohammad Ahmadi
Asia Pacific University of Technology and Innovation
21 PUBLICATIONS **126** CITATIONS
SEE PROFILE

An Introduction to Docker and Analysis of its Performance

Babak Bashari Rad, Harrison John Bhatti, Mohammad Ahmadi
Asia Pacific University of Technology and Innovation
Technology Park Malaysia, Kuala Lumpur, Malaysia

Summary
Docker provide some facilities, which are useful for developers and administrators. It is an open platform can be used for building, distributing, and running applications in a portable, lightweight runtime and packaging tool, known as Docker Engine. It also provide Docker Hub, which is a cloud service for sharing applications. Costs can be reduced by replacing traditional virtual machine with docker container. It excellently reduces the cost of re-building the cloud development platform.

Key words:
Docker, Docker Container, Virtual Machine, Virtualization, Cloud Computing.

1. Introduction

Docker is an open source platform that run applications and makes the process easier to develop, distribute. The applications that are built in the docker are packaged with all the supporting dependencies into a standard form called a container. These containers keep running in an isolated way on top of the operating system's kernel. The extra layer of abstraction might effect in terms of performance [1].
Even thou, the technologies of the container have been around for over 10 years, but docker, a generally new hopeful is right now a standout amongst the best innovations, since it accompanies new capacities that prior technologies did not have. Initially, it gives the facility to create and control containers. Besides that, applications can easily be packed into lightweight docker containers by the developer. These virtualized applications can easily be worked anywhere without any alteration. Moreover, docker can convey more virtual situations than different innovations, on the same equipment. To wrap things up, docker can easily coordinate with third-party instruments, which help to easily deploy and manage docker containers. Docker containers can easily be deployed into the cloud-based environment [2].
This paper is a review on technology of docker, and will analyse its performance by a systematic literature review. The article is organised as follow. Next section will introduce the technology of docker. In Section 3, a more detailed description of docker and its components will be presented. Section 4 briefly compare technology of Virtual Machine and Docker. Sections 5 and 6 will discuss the advantages and disadvantages of docker container, respectively. In Section 6 and 7, we briefly review few recent researches on measuring the performance of Docker and compare it with other container technologies. Finally, in section 9 and 10, features in virtual machines and containers will be briefly summarised, following with a short summary of the paper.

2. Docker

Docker provides a facility to automate the applications when they are deployed into Containers. In a Container environment where the applications are virtualized and executed, docker adds up an extra layer of deployment engine on top of it. The way that docker is designed is to give a quick and a lightweight environment where code can be run efficiently and moreover it provides an extra facility of the proficient work process to take the code from the computer for testing before production [9]. Russell (2015) confirms that, as quick as it is possible docker allows you to test your code and deploy it into the production environment [6]. Turnbull (2014) concludes by saying that docker is amazingly simple [9]. Certainly, you can begin with a docker with a simple configuration system, a docker binary with Linux kernel.

3. Docker Inside

There are four main internal components of docker, including Docker Client and Server, Docker Images, Docker Registries, and Docker Containers. These components will be explained in details in the following sections.

3.1 Docker Client and Server

Docker can be explained as a client and server based application, as depicted in Figure 1.
The docker server gets the request from the docker client and then process it accordingly. The complete RESTful (Representational state transfer) API and a command line client binary are shipped by docker. Docker daemon/server and docker client can be run on the same machine or a local docker client can be connected with a remote server or daemon, which is running on another machine [9].

Manuscript received March 5, 2017
Manuscript revised March 20, 2017

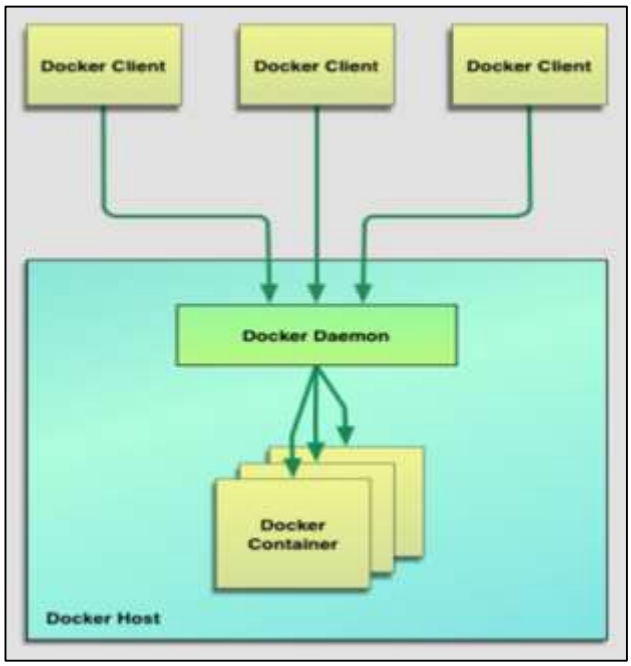

Fig. 1 Docker architecture [9].

3.2 Docker Images

There are two methods to build an image. The first one is to build an image by using a read-only template. The foundation of every image is a base image. Operating system images are basically the base images, such as Ubuntu 14.04 LTS, or Fedora 20. The images of operating system create a container with an ability of complete running OS. Base image can also be created from the scratch. Required applications can be added to the base image by modifying it, but it is necessary to build a new image. The process of building a new image is called "committing a change". The second method is to create a docker file. The docker file contains a list of instructions when "Docker build" command is run from the bash terminal it follows all the instructions given in the docker file and builds an image. This is an automated way of building an image.

3.3 Docker Registries

Docker images are placed in docker registries. It works correspondingly to source code repositories where images can be pushed or pulled from a single source. There are two types of registries, public and private. Docker Hub is called a public registry where everyone can pull available images and push their own images without creating an image from the scratch. Images can be distributed to a particular area (public or private) by using docker hub feature.

3.4 Docker Containers

Docker image creates a docker container. Containers hold the whole kit required for an application, so the application can be run in an isolated way. For example, suppose there is an image of Ubuntu OS with SQL SERVER, when this image is run with docker run command, then a container will be created and SQL SERVER will be running on Ubuntu OS.

4. Virtual Machine vs. Docker

Virtualization is an old concept, which has been in used in cloud computing, after IaaS has been accepted as a crucial technique for system constitution, resource provisioning, and multi-tenancy. Virtualized resources play the main role in solving the problems using the core technique of cloud computing. The Figure 2 shows the architecture of the virtual machine.

Fig. 2 Virtual Machine architecture [11].

Hypervisor is lying between host and guest operating systems. It is a virtual platform and it handles more than one operating system in the server. It works between the operating system and CPU. The virtualization divides it into two segments: the first one is Para-Virtualization and the second one is Full Virtualization [3]. Figure 3 depicts the architecture of the Docker Container.

Linux containers are managed by the docker tool and it is used as a method of operating system level virtualization. Figure 3 shows that in single control host there are many Linux containers, which are isolated. Resources such as Network, Memory, CPU, and Block I/O are allocated by Linux kernel and it also deals with cgroups without starting virtualization machine [8].

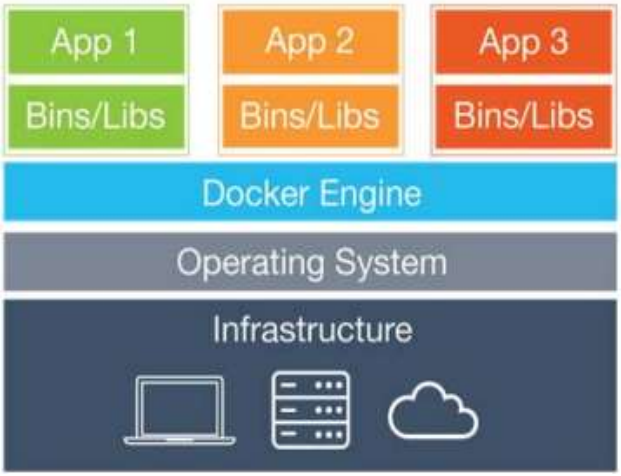

Fig. 3 Docker Container architecture [11].

According to Waldspurger (2002), in the Linux containers, an architecture is to manage CPU and distribute its resources more proficiently. In any example of Hyper-V or VMWare, because of overhead incurred, it is not easy to run more than ten virtual machines [13]. Up to a great extent, this issue has been solved by the containers. Containers only utilize those resources, which are needed for the services or applications. Therefore, on a weak configured machine, above 50 requests of the containers can be executed.

For example, suppose an organisation provides email security services. The major functions of these services are to check emails for viruses, spam, and malware. Moreover, it could manage to transfer messages to the agent, logs and report delivery failure if the product is installed in the cloud [10]. Mostly in these cases, there is no use of any associated dependencies or OS level libraries or any kernel data structure. Therefore, it is worthwhile to containerized every component by sandboxing them utilizing OpenVZ or Docker instead of having virtual machines.

In many enterprises, virtual machines are used to perform element testing. In this process, a lot of CPU resources and memory space are consumed. Whereas, container technology provides a guarantee to their users that excess of a workload would not affect the efficiency of the resources. The container takes less time for installation as compared to virtual machines, so the adaptability of containers is much higher than VMs.

Furthermore, both Docker and OpenVZ have been under great examination in terms of their security aspects. When isolation is reduced, it directly affects the security, which also decreases rapidly. Root users of Linux can easily get access to containers as containers also use the same kernel and operating system. The isolation of docker is not as strong as a virtual machine, even though docker isolates the application, which is running in the docker container from its primary host. Additionally, it is possible that some of the applications would not be able to run in a containerized technology and they need to run on a different operating system.

5. Advantages of Docker Container

The demand and the advancement of Linux containers can be seen in the last few years. Docker has become popular very quickly, because of the benefits provided by docker container. The main advantages of docker are speed, portability, scalability, rapid delivery, and density.

5.1 Speed

Speed is one of the most exceedingly touted advantages of Containers. When the benefits of using docker are highlighted, it would be incredible not to mention about the speed of docker in the conversation (Chavis & Architect, 2015). The time required to build a container is very fast because they are really small. Development, testing, and deployment can be done faster as containers are small. Containers can be pushed for testing once they have been built and then from there, on to the production environment [12].

5.2 Portability

Those applications that are built inside docker containers are extremely portable. These portable applications can easily be moved as a single element and the performance remains the same [12].

5.3 Scalability

Docker has the ability that it can be deployed in several physical servers, data servers, and cloud platforms. It can also be run on every Linux machine. Containers can easily be moved from a cloud environment to local host and from there back to cloud again at a fast pace. Adjustments can easily be done; the scale can simply be adjusted by the user according to the need [5].

5.4 Rapid Delivery

The format of a Docker Containers is standardized so programmers do not have to stress over one another's tasks. The responsibility of the administrator is to deploy and maintain the server with containers, whereas the responsibility of the programmer is to look after the applications inside the docker container. Containers can work in every environment as they have all the required dependencies embedded within the applications and they are all tested [12]. Docker provides a reliable, consistent, and improved environment, so predictable results can be

achieved when codes are moved between development, test and production systems (Chavis & Architect, 2015).

5.5 Density

Docker uses the resources that are available more efficiently because it does not use a hypervisor. This is the reason that more containers can be run on a single host as compared to virtual machines. The performance of a Docker Containers is higher because of higher density and no overhead wastage of resources [5].

6. Disadvantages of Docker Container

There are some drawbacks of docker containers, which are listed below [1, 4]:

- Complete virtualization is not provided by a docker because it depends on the Linux kernel, which is provided by the local host.
- Currently, docker does not run on older machines. It only supports 64-bit local machines.
- The complete virtualized environment must be provided by the docker container for Windows and Mac machines. Even though the *boot2docker* tool fills this gap, but still, it should be checked whether it makes obstructions to acceptance by users of these systems or the integration and performance with the host machine's operating system are adequate [4].
- It is necessary that the possibility of security issues should be evaluated. Building off trusting binaries could be made easier by digitally signing docker images, for future support.
- An important concern is to check if the teaching community or scientific researcher will significantly think of adopting docker.

7. Docker Performance

Seo et al. (2014) used two servers with the same configuration in the cloud environment. One server was used for docker and the other one was for an Open Stack platform for KVM by means of a virtualization tool [8].
According to him, a VM works independently. This factor make it easy to apply and manage the policy of network, security, user, and the system. However, docker does not contain a guest operating system. Therefore, it takes very little time in distributing and gathering images. The boot time is also very short. These are the main advantages of utilizing Docker Cloud as compared with VM Cloud.
Scheepers (2014) compares LXC and Xen virtualization technologies to benchmark some applications [7]. He explains that Xen would be a better choice in the sense of equally distributing resources, performance is not dependent on the other tasks, and it is executed on the same machine. However, LXC is much better in the sense of getting most of the hardware resources or for the execution of smaller isolated processes. In private and dot clouds, LXC is a better option.
Felter et al. (2014) evaluate the performance of three different environments, Native, Docker, and KVM [3]. He clarifies that containers and VMs are both mature innovation that has profited from last 10 years of incremental equipment and programming enhancements. According to this research, docker is equivalent to or surpasses KVM execution for each situation they tried. Their outcomes demonstrate that both KVM and docker present irrelevant overhead for CPU and memory execution. It has also been shown that the overall performance of docker is better than the Local Host, as the applications were executed and responded faster than in Local Host. Moreover, fewer hardware resources were used in docker container to perform the tasks.
Docker is really a future demanding technology. As users and developers would know more about the docker and its capabilities then they would consider replacing traditional virtualization with docker technology. Docker provides many simple and useful features. To get the best performance and results, it is highly recommended to move up from the default configuration. Containers provide advanced density, better performance, scalability, and usability as compared with traditional virtualization because containers smartly utilize its resources, which reduce the chance of unnecessary overhead. Containers are better in performance than virtual machine, because containers take less start-up time. Docker has removed the biggest issue of "dependency". Now containers have all of their required dependencies, which help containers to be properly built, and to execute them in any docker environment. An additional layer of isolation is provided by the container, which increases the containers' security. Docker is not as insecure as people normally think, but it provides a complete protection.

8. Docker vs. other Container Technology

In this section, the performance of application virtualization and the performance of the docker container will be discussed, and the evaluation of other containerize technology will be compared and reviewed. Seo et al. (2014) summarize that there is no guest OS of docker in the cloud, so the storage and the wastage of CPU resources are less [8]. The images are not disturbed; boot time is faster and the time of generating the images is short. These are the benefits of docker cloud in comparison with VM Cloud. They used two similar servers with the same configuration in the cloud environment. One server was used for docker and the other one was for an Open Stack platform for KVM

by means of a virtualization tool. Ubuntu Server was used as a base platform [8].

To calculate the approximate boot-time, 20 images were generated on each server and boot time was checked. Figure 4 shows that the boot time of docker is lesser than the boot time of KVM. Docker uses the Host OS, whereas KVM uses Guest OS. Thus, the boot time of docker is shorter than the boot time of KVM.

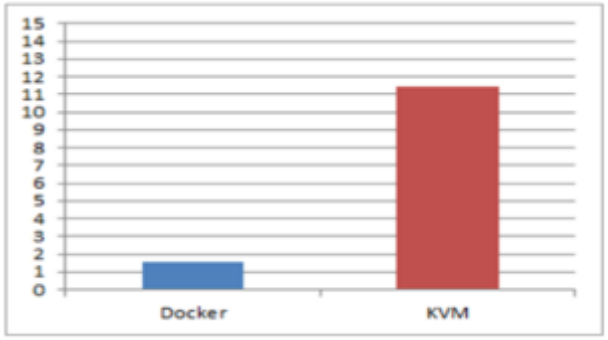

Fig. 4 Docker vs KVM Average boot time [8].

To calculate the operational speed, python language was used. Figure 5 shows that operation speed of 100,000 is averagely around 4.5s. To measure the operation speed, they obtain the average process time and standard deviation, by repeating the same process 100 times on docker and VM.

Fig. 5 CPU Calculation Performance [8].

Figure 5 shows the calculation speed of docker is slightly faster than the calculation speed of the VM [7].

Seo et al. (2014) concluded that VM works independently [8]. This is one of the reasons that it is easy to apply and manage the policy of network, security, user, and the system. However, docker does not contain a guest operating System. Therefore, it takes very less time in distributing and gathering images. Its boot time is also very short. These are the main advantages of utilizing docker cloud as compared with VM Cloud.

Scheepers (2014) compares LXC and Xen virtualization technologies to benchmark some applications [7]. For this purpose, Scheepers uses two servers Core OS 324.3.0 and XenServer 6.2 with docker version 0.11.1. The configuration of these systems is RAM 4GB, CPU Intel Xeon Quad core and the virtualization support is Intel VT-X. The base operating system is Ubuntu 12.04 and containers will run on both machines. 2GB of memory is allotted to the first virtual machine and Apache 2.2, WordPress 3.9 and PHP 5.3. This was used as an application Server. 1GB of memory is used by the second virtual machine with MYSQL Database 5.5. This database was filled by the WordPress sample contents. This machine was used as a database Server. JMeter was used as a benchmarked tool.

Figure 6 shows that LXC experienced less overhead as compared to Xen when the SELECT query was run. The focus on running this benchmark process is to see the utilization of the CPU and the performance of the Network speed because these are the main resources consumed in this test.

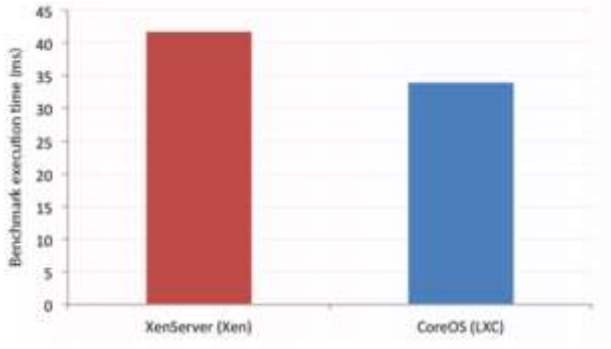

Fig. 6 Time in millisecond to complete a one SQL select query [7].

Figure 7 shows that in Xen setup, it took 16 seconds to accomplish when the INSERT query was run in the database, whereas in LXC setup it took longer–around 335 seconds. This reveals the inability of LXC container to isolate resources efficiently.

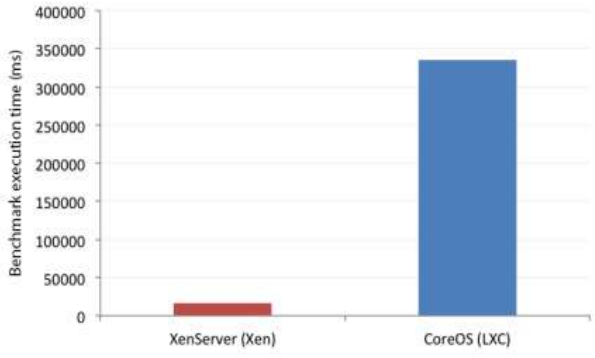

Fig. 7 Time in millisecond to complete 10,000 SQL INSERT queries [7].

Scheepers (2014) concludes that Xen would be a better choice in the sense of equally distributing resources, performance is not dependent on the other tasks, and it is executed on the same machine [7]. However, LXC is much better in the sense of utilizing most of the hardware resources or the execution of smaller isolated processes. In private and dot clouds, LXC is a better option.

Felter et al. (2014) evaluated the performance of three different environments, Native, Docker, and KVM [3]. Overhead issues are also highlighted in their research. Scenarios were investigated where more than one hardware resource was completely utilized. To perform the tests, they used an IBM x3650 M4 server, 16 core processors of Xeon E5-2665, Two Intel Sandy Bridge-EP of 2.4 - 3.0 GHz and 256 GB of RAM. To make a non-uniform memory access, two processors were linked together with QPI link. Cloud providers also use this kind of similar Server. The base operating system was Ubuntu 13.10, docker version 1.0, Linux kernel 3.11.0, libvirt version 1.1.1 and QEMU 1.5.0. This Figure 8 shows that the average size of 1 MB was used for I/O, little over 60 seconds by measuring the performance of sequential read and write. In this case, slight overhead can be seen by Docker and KVM. In other cases, KVM has almost a four times performance difference.

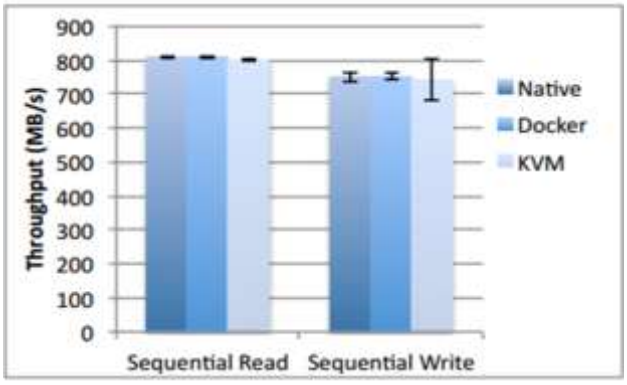

Fig. 8 Sequential I/O throughput (MB/sec) [3].

Figure 9 demonstrates the execution of irregularly read, write and mixed workloads utilizing a 4 kB square size and simultaneousness of 128, which we tentatively decided gives the greatest execution to this specific SSD. As we would expect, docker acquaints no overhead contrasted and Linux, however, KVM conveys just have the same number of IOPS since every I/O operation must experience QEMU. While the VM's supreme execution is still very high, it utilizes more CPU cycles per I/O operation, leaving less CPU accessible for application work.

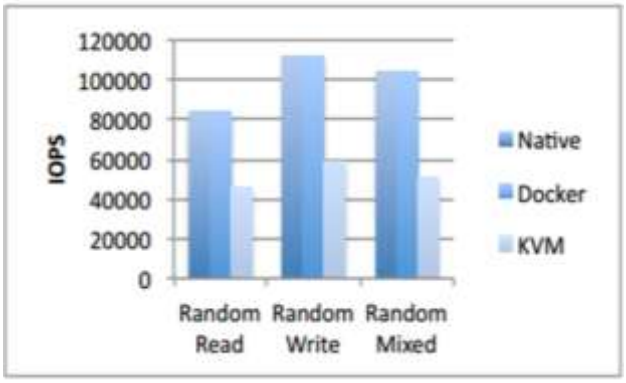

Fig. 9 Random I/O throughput (IOPS) [3].

Felter et al. (2014) conclude that containers and VMs are both mature innovations that have profited from the last 10 years of incremental equipment and programming enhancements [3]. When all is said and done, docker is equivalent to or surpasses KVM execution for each situation we tried. Our outcomes demonstrate that both KVM and docker present irrelevant overhead for CPU and memory execution.

To conclude these past works, regardless of utilizing distinctive techniques and having diverse centres, one thing is common that is measured and comparing the performance of applications and different types of containerized and virtualized technology.

9. Virtual Machines vs. Containers

Table 1 compare features of different containerized and virtual machine technologies. Virtual machine uses an extra layer between the host operating system and guest operating system. This layer is known as a Hypervisor. Whereas docker adds up an extra layer between host operating systems and where the applications are virtualized and executed, which is known as a Docker Engine. As docker does not use any guest operating system that makes a big difference in performance between a docker container and a virtual machine technology. In Table 1, the performances of applications running in different containers and virtual machines are also briefly compared.

As it is given in the table above, according to Seo et al. (2014) the docker performance is better than KVM, in terms of boot time and calculation speed [8], whereas Felter et al. (2014) proves that there is no difference of wastage of resources (overhead) between Docker and KVM but there is a noticeable difference in execution, as KVM is faster than Docker [3]. Scheepers (2014) found out that LXC takes a longer time to accomplish tasks, whereas Xen Server takes less time [7]. LXC is better in the sense of fewer wasted resources while Xen is better in the sense of equally distributing resources.

Table 1: Comparison Table based on Different Virtual Machines and Containerized Technology

Seo et al. (2014) [8]		Scheepers (2014) [7]		Felter et al. (2014) [3]		
Docker	**KVM**	**XenServer (Xen)**	**CoreOS (LXC)**	**Native**	**Docker**	**KVM**
Boot Time short	Boot time long	More overhead (wastage of resources)	Less overhead (wastage of resources)	Overhead (wastage of resources)	Slightly less overhead than Native	Slightly less than Native and Docker
Calculation Speed is faster	Calculation Speed is Slower	Less time to accomplish request	Longer time to accomplish request	Slow Execution equal to Docker	Slow Execution equal to Native	Fast Execution
No Guest OS	Works Independently	Better in sense of equally distributing resources	Better in sense of executing isolated processes	-	Mature Innovation	Mature Innovation

10. Summary

Docker automates the applications when they are containerized. An extra layer of docker engine is added to the host operating system. The performance of docker is faster than virtual machines as it has no guest operating system and less resource overhead.

References

[1] Boettiger, C. (2015). An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, 49(1), 71-79.

[2] Bui, T. (2015). Analysis of docker security. arXiv preprint arXiv:1501.02967.

[3] Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2014). An updated performance comparison of virtual machines and linux containers. technology, 28, 32.

[4] Harji, A. S., Buhr, P. A., & Brecht, T. (2013). Our troubles with Linux Kernel upgrades and why you should care. ACM SIGOPS Operating Systems Review, 47(2), 66-72.

[5] Joy, A. M. (2015). Performance comparison between Linux containers and virtual machines. Paper presented at the Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in.

[6] Russell, B. (2015). Passive Benchmarking with docker LXC, KVM & OpenStack.

[7] Scheepers, M. J. (2014). Virtualization and containerization of application infrastructure: A comparison.

[8] Seo, K.-T., Hwang, H.-S., Moon, I.-Y., Kwon, O.-Y., & Kim, B.-J. (2014). Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud.

[9] Turnbull, J. (2014). The Docker Book: Containerization is the new virtualization.

[10] Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. Knowledge and Data Engineering, IEEE Transactions on, 16(9), 1128-1142.

[11] Varghese, B., Subba, L. T., Thai, L., & Barker, A. (2016). Container-Based Cloud Virtual Machine Benchmarking. arXiv preprint arXiv:1601.03872.

[12] Vase, T. (2015). Advantages of Docker.

[13] Waldspurger, C. A. (2002). Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review, 36(SI), 181-194.

[14] ACM SIGOPS Operating Systems Review. 36 p.181-194.

Dr Babak Bashari Rad received his B.Sc. of Computer Engineering (Software) in 1996 and M.Sc. of Computer Engineering (Artificial Intelligence and Robotics) in 2001 from University of Shiraz and Ph.D. of Computer Science in 2013 from University Technology of Malaysia. Currently, he is the Programme Leader of postgraduate studies and senior lecturer in the School of Computing, Asia Pacific University of Technology and Innovation (APU), Kuala Lumpur Malaysia. His main research interest covers a broad range of various areas in computer science and information technology including Information Security, Malware Detection, Machine Learning, Artificial Intelligence, Image Processing, Robotics, Cloud Computing, Big Data, and other related fields.

Harrison John Bhatti received his Bachelors of Science in Computer Science (BCS) degree in 2003 and M.Sc. of Information Technology Management in the field of Cloud Computing and Virtualization in 2016 from Asia Pacific University of Technology and Innovation (APU), Kuala Lumpur in Collaboration with Staffordshire University, UK. Harrison John is currently doing his second Masters of Engineering in Industrial Management and Innovation from University of Halmstad, Sweden. His core research areas are Cloud Computing, Virtualization, Docker Container and Strategic Planning and Innovation.

Dr Mohammad Ahmadi received his PhD in computer science with specialization in multimedia computing from UPM university of Malaysia in 2014, M.Sc. in IT engineering from AmirKabir Poly-technique University of Tehran in 2007, Iran, and B.Sc. in computer software engineering from Shiraz Azad University, Iran in 2003.

He is a Senior Lecturer in faculty of Computing, Technology and Engineering of Asia Pacific University of Technology and Innovation, Malaysia. He used to be lecturer in different universities in Iran, and He has experienced Casual Lecturer at Western Sydney University, Australia. Dr. Ahmadi has published several papers on high ranked journals such as Emerald or IJST. His main research interest covers a broad range of various areas in computer science and information technology including serious games, multimedia, cloud computing, mobile applications, e-learning, and computer graphics.