Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.8 No.S2, 2019, pp.28-30
© The Research Publication, www.trp.org.in
DOI: https://doi.org/10.51983/ajcst-2019.8.S2.2030

Performance Comparison between VM Based Webserver and Docker Container Webserver

A. Anand[1] and A. Nisha Jebaseeli[2]
[1]Digital Transformation Group, Bharat Heavy Electricals Ltd, Tiruchirappalli Division, Tamil Nadu, India
[2]Assistant Professor & Head, Department of Computer Science,
Bharathidasan University Constituent College, Lalgudi, Tamil Nadu, India
E-Mail: anand_visuvasam@yahoo.com, nishamarcia@gmail.com

(Received 7 January 2019; Revised 23 January 2019; Accepted 13 February 2019; Available online 20 February 2019)

***Abstract* - Cloud computing is a type of Internet-based computing that provides shared computer processing resources, services and data to computers on demand. It offers an innovative business model for organizations to adopt IT services at a reduced cost with increased reliability and scalability. Virtualisation is one of backbone technology of cloud computing. But today, container based technology especially Docker offering better performance than Virtual Machine. It is famous for its light weight operation and better scaling. But still it is lagging in Disk I/O and network bandwidth intensive applications. So it is important to analyse and compare various performance parameters of VMs and Docker Images before implementation. Main Parameters will be CPU, Memory, Disk Utilization and Network Bandwidth. In this research paper, we compare performance metrics between Webserver deployed in Virtual machine and Docker webserver.**

***Keywords:* Docker Container, Virtual Machines, Performance Metrics, Web server, Cloud Computing**

I. INTRODUCTION

Cloud computing is an on-demand resource model to deliver high performance and reliability. Three main forms of Cloud computing are private cloud, public cloud, and hybrid clouds. Public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as virtual machines (VMs), applications or storage, available to the general public over the internet. Public cloud services may be free or offered on a pay-per-usage model. Private cloud is a particular model of cloud computing that involves a distinct and secure cloud based environment in which only the specified client can operate. Hybrid cloud includes multiple private cloud or public cloud platforms provided by multiple cloud providers. Hybrid cloud can dynamically increase the response and processing capacity of the Web applications by dynamically extending the Web applications to the nearly unlimited public cloud resources.

Virtualisation technology is key concept in cloud computing. Virtualization provides a layer of abstraction between the hardware and the software. Hardware or platform virtualization refers to the creation of a virtual machine (VM) that acts like a real computer with an operating system. The main difference from the tradition computer is that it allows definition of multiple VMs with different operating systems over the same hardware. The host machine is the actual machine on which the virtualization takes place and the guest machine is the virtual machine created by hypervisor [4] (Virtual Machine Manager). In virtual machines, isolation is achieved at machine level and each virtual machine runs its own operating system. Interaction between hardware and operating system will be done through hypervisors. It is computer software, firmware or hardware that creates and runs virtual machines.

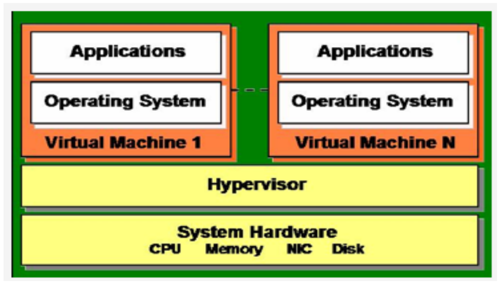


Fig. 1 Virtual Machines

Each virtual machine appears to have the host's processor, memory and other resources all for itself. However, it is actually controlling the host processor and other resources, allocating what is needed to each operating system and making sure that the guest machines (virtual machines) can't disrupt each other.

Different hypervisors available in market are VMWare ESX, XEN, KVM and Hyper-V etc.,

II. CONTAINER TECHNOLOGY

As technology evolves, container-based technology [2] has gained significant traction in the last years due to its near native performance for application execution while featuring application isolation. Unlike virtual machines, containers execute directly on the host OS, sharing the kernel with other containers. A container is a uniform structure in which any application can be stored, transported and run. It is named for and often compared to the standardised intermodal containers used in the shipping industry for efficient transportation. A container encapsulates an application with its own operating environment. It can be placed on any host machine without special configuration, removing the issue of dependencies. It requires minimum amount of resources to perform the task.

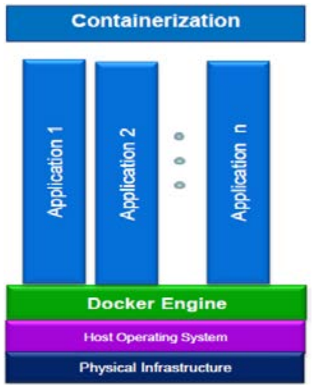


Fig. 2 Container Virtualisation

Different container technologies available in market are OpenVZ, LXC, Rkt and Docker etc.

III. DOCKER

Docker [7][9] is a tool which renders the lightweight virtualization at system level through extending a common container format approach on Linux called Linux Containers (LXC). It employs c groups [5] (Linux control groups), for resource allocation and isolation. LXC limits an application to a specific set of resources and it enables Docker to share the host OS resources. It automates a portable, lightweight, self-sufficient container deployment of any application that will scuttle virtually anywhere. Such Docker containers can enclose any payload, and will run coherently on and between virtually any servers.

The Docker platform is divided into the Docker Engine, which supports the runtime and execution of containers, and the Docker Registry, which provides the hosting and delivery of a repository of Docker images. Each container provides a namespace, isolated environment for execution. Docker exploits filesystem layering, as well as specific features of the Linux kernel to make all of these possible.

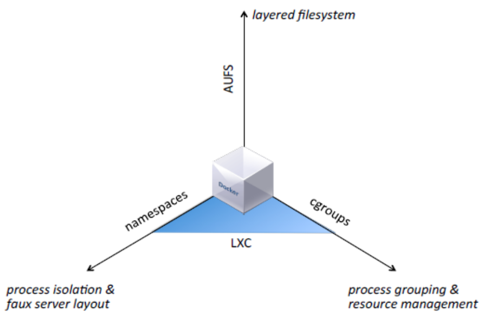

layered filesystem
AUFS
Docker
namespaces
cgroups
LXC
process isolation & faux server layout
process grouping & resource management

Fig.3 Docker Platform

Namespaces are the mechanism by which each Docker container is isolated from the host and other containers. There are many different namespaces that LXC supports, but perhaps the two most significant ones are the pid [5] and net namespaces [5]. The pid namespace is responsible for giving each container its own isolated environment for processes. A given container can only see and send signals to the processes that are running within the same container. In addition, the net namespace allows different containers to have what appears to be distinct network interfaces, thereby permitting two containers to simultaneously bind to the same port. AUFS, (Another Union Filesystem) [8] is a file system used by Docker that amalgamates a collection of different file systems and directories into a single logical file system.

IV. INSIDE DOCKER

Four main internal components of docker [10] are Client and Server, Docker Images, Docker Registries, and Docker Containers. The docker server gets the request from the docker client and then processes 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. The foundation of every image is a base image. Operating system images are basically the base images. The images of operating system create a container with an ability of complete running OS. Docker images are placed in docker registries. 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. 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.

V. EXPERIMENTAL SETUP

In this research, Suse Openstack private cloud setup was used for deploying VM and Docker. RHEL 7 was installed and used as Base OS for both Virtual Machine and Docker image. JSP based Model Supplier registration application was developed and deployed in Webserver. Apache Tomcat was used as web application server. It was installed and configured in both VM and Docker. JMeter was used to generate web loads. In order to test the different scenarios of deploying web services, JMeter was used as a generation and testing tool. JMeter comes with a graphical server performance dashboard. JMeter simulates a group of users sending requests to a target server then statistics will be provided indicating the performance of that specific target server.

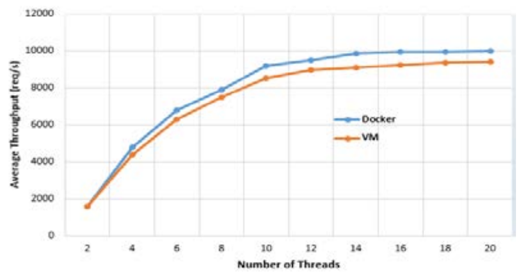
VI. PERFORMANCE METRICS

In order to effectively analyse the performance between Webserver deployed in VMWare VM and Docker webserver [6], following metrics [1] [3] are considered in this test:

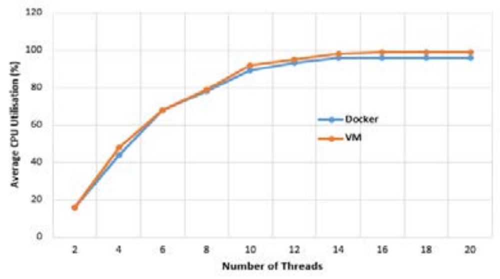
1. *Server Throughput (requests/sec):* It is important to measure server performance with respect of throughput at different request rates. It is a rate at which we are able to retrieve the web documents with the GET method of HTTP requests sent.
2. *Response Time (millisecond):* It is essential to measure the time elapsed from sending the request until a response is received. The response time is measured in milliseconds. Both response time and throughput are measured using the measurement tool JMeter.
3. *CPU Utilization (percent):* Average CPU Utilization is Amount of CPU utilisation due to handling incoming number of requests.

VII. RESULTS AND DISCUSSION

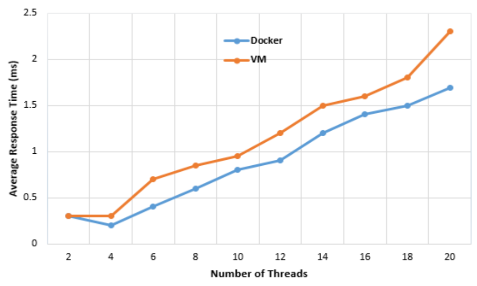
The test was done for 20 seconds duration. The number of threads used in this test is varying from 1 to 20. JMeter is used to generate request and receive response from webserver. Performance metrics were captured and plotted in curve. Average number of successful requests per second handled by server is plotted in fig 4(a). It is observed that the throughput increases with increase of number of threads. The throughput saturates approximately 10,000 requests per second in Docker Webserver. Whereas for VM based webserver maximum throughput is obtained approximately 9, 400 requests per second. The CPU utilisation (Fig 4 (b)) in Docker webserver is reaching 96% when handling request of 20 threads. But in VM based Webserver CPU Utilisation is reaching nearly 100% when handling request of 16 threads. Fig 4(c) shows Docker webserver has lower response time when comparing with VM based webserver means that Docker webserver is faster in handling incoming requests.



(a)



(b)



(c)

Fig. 4 Performance metrics

VIII. CONCLUSION

In this paper, it was demonstrated that Docker web server outperform VM based webserver in all performance metrics. The performance of docker is faster than virtual machines as it has no guest operating system and less resource overhead.

REFERENCES

[1] Tasneem Salah, M. Jamal Zemerly, and Chan Yeob Yeun "Performance comparison between container-based and VM-based services", *IEEE International conference on Innovations in Clouds, Internet and Networks, Paris*, pp. 185-190, March 2017

[2] Germán Moltó, Miguel Caballer, and Alfonso Pérez "Coherent Application Delivery on Hybrid Distributed Computing Infrastructures of Virtual Machines and Docker Containers", *IEEE International conference on Parallel, Distributed and Network-based Processing (PDP), St. Petersburg, Russia*, pp. 486-490, March 2017.

[3] Janki Bhimani, Zhengyu Yang, and Miriam Leeser "Accelerating big data applications using lightweight virtualization framework on enterprise cloud", *IEEE International conference on High Performance Extreme Computing Conference (HPEC), IEEE, Waltham, MA, USA*, pp. 1-8, 2017.

[4] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio "An updated performance comparison of virtual machines and Linux containers", *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 171-172, 2015.

[5] Joris Claassen, Ralph Koning, and Paola Grosso "Linux containers networking: Performance and scalability of kernel modules" *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 713-717.

[6] Yunchun Li, and Yumeng Xia "Auto-scaling web applications in hybrid cloud based on docker" *5th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 75-79, 2016.

[7] Nitin Naik "Building a virtual system of systems using docker swarm in multiple clouds" *IEEE International Symposium on Systems Engineering (ISSE)*, pp.1-3, 2016.

[8] Theo Combe, Antony Martin, and Roberto Di Pietro "To Docker or Not to Docker: A Security Perspective" *IEEE Cloud Computing Year: 2016*, Vol. 3, No. 5, pp. 54-62, October 2016.

[9] Bin Xie, Guanyi Sun, and Guo Ma "Docker based overlay network performance evaluation in large scale streaming system" *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 366-369, 2016.

[10] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi "An Introduction to Docker and Analysis of its Performance" *IJCSNS International Journal of Computer Science and Network Security*, Vol. 17, No. 3, pp. 228-235, March 2017.