Proceedings of the International Conference on Communication and Electronics Systems (ICCES 2018)
IEEE Xplore Part Number:CFP18AWO-ART; ISBN:978-1-5386-4765-3

Performance Evaluation for Deploying Docker Containers On Baremetal and Virtual Machine

Ashish Lingayat
Computer Engineering
MIT Academy of Engineering
Pune, India
ashishvijaylingayat@gmail.com

Ranjana R. Badre
Computer Engineering
MIT Academy of Engineering
Pune, India
rrbadre@comp.maepune.ac.in

Anil Kumar Gupta
HPC Infrastructure and Ecosystems
C-DAC Innovation Park
Pune, India
akgupta5592@gmail.com

***Abstract*—The current virtualization technology induces an overhead in their performance. The benefits of Linux containers led to their development and increasing its usage among administrators and developers. Docker container offers an isolated space for running the application in the containerized environment. Containers are known for starting applications in quicker time as compared to those running on baremetal or virtualized environment. For calculating the startup time of containerized applications, we are using a benchmark based on Docker containers. This benchmark is capable of calculating the startup time of Docker application running services. The services can be Apache web server, Redis, PostgreSQL, and many others. The proposed system calculates the startup time for Docker containers running the Apache web server in baremetal, and virtual machine (KVM). The test will let the users know about the performance of Docker applications in both the environments. The analysis will help them in deciding the environment for working with Docker application to obtain maximum performance. The experimental setup of the proposed system shows that running containerized application on baremetal will improve the startup time than a virtual machine. The performance on baremetal is enhanced by about 50% than a virtual machine.**

***Index Terms*—Linux containers, Benchmark, Docker, virtualization.**

I. Introduction

Docker is a container based platform developed to create and run applications directly. It is an open source platform for building, shipping and running distributed system [1]. Docker containers have proven very beneficial for running applications in containerized environments. Though it is not a complete alternative to the virtual machine, still they provide isolation for other container applications [2] [3]. Dockers are immensely used for development, testing, and deployment of applications [4]. Docker applications are packaged together for guaranteed testing of applications across other platforms by encapsulating the dependencies, libraries, binaries or other required pieces of software in a container.

Benefits of using Docker containers:

- Services are scaled by adding or removing some containers.
- The density of the Docker container can be increased to handle more workloads.
- They are built very easily and quickly because their deployment time is very less.
- The efficiency of Docker containers given resource utilization is very high due to limitation and other resource management strategies.

The startup time of container application is essential for scaling, rescheduling, upgrades, building, and testing [5]. For calculating the startup time of Docker container, we are using benchmark. The benchmark is lightweight and calculates the time for launching the Docker container. The time estimated is based on the time consumed by a Docker container by deploying a particular service. For benchmarking, we have used standard Docker container application available on the official Docker repository. The startup time of the container is also affected by the app being launched. In this paper, we have used the Apache web server application with standard implementation with the single web page.

II. Related work

Wes Felter et al. [6] shows the performance comparison on baremetal, virtual machine, and Linux containers. Their result help in identifying the performance factors responsible. They have used numerous evaluation platforms and applications to obtain useful results. Their results show that Linux containers outperform virtual machine whereas baremetal proves tends to be faster than Linux containers.

Jyoti Shetty et al. [7] show performance evaluation for native, virtual machine, and Docker containers. The performance measures were based on CPU, memory, disk, and system. Their assessment shows that Docker containers are better than the virtual machine regarding performs but lack isolation.

Ashish Lingayat et al. [8] proposed a comparative study on the performance evaluation based on various environments. The environments were based on baremetal, virtual machine, Linux containers, and OpenStack cloud for evaluating CPU speed, network, disk, memory, Apache web server. The analysis shows the performance variation between Linux containers and virtual machine caused primarily by the architecture of the systems. The research indicates that Linux containers are beneficial than virtual machine and OpenStack cloud.

Ali Babar et al. [9] calculate the performance of systems based on the host operating system, virtual machine, and Linux containers. Their evaluation shows that Linux containers are

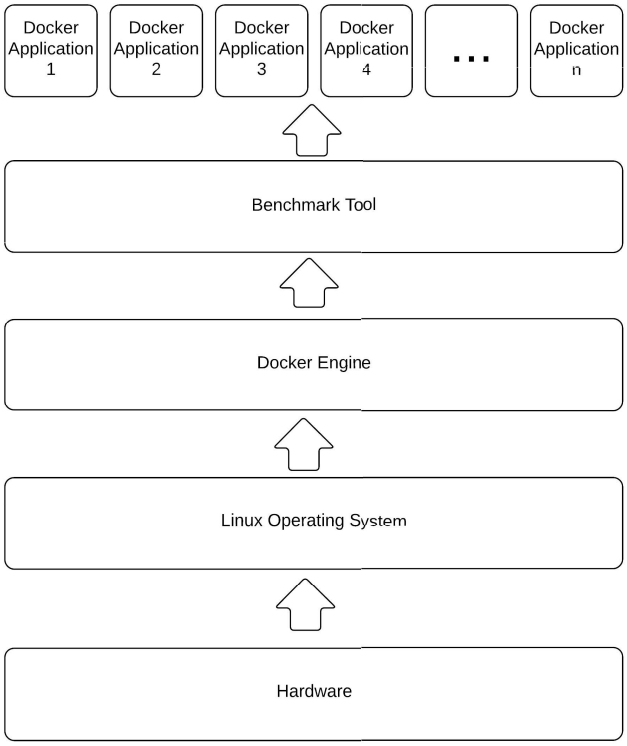

Fig. 1. Architecture of proposed system.

a better option over virtual machine due to the increased performance.

Yoji Yamato [10] measured the startup time for starting services in baremetal, Docker containers and virtual machine deployed in an OpenStack cloud environment. The results obtained showed that Docker containers start applications in shorted time against the virtual machine.

Ashish Lingayat et al. [11] enlists the various projects in the OpenStack cloud that are under development for improving the performance and obtaining better results. These projects help in deploying the application in the containerized environment by using OpenStack cloud features.

According to M Fisher [12] lmbench and BYTEmark Native Mode Benchmark ver. 2(nbench) open source benchmarks to measure the performance of each environment.

III. Proposed work

The literature survey shows the absence of evaluating the startup time for applications on the baremetal and virtual machine for deploying Linux containers. The proposed system is developed for calculating the startup time for running containerized applications using Docker containers. The startup time is computed using the benchmark tool. This work will prove useful for the administrators and developers in deciding the platform for running the applications.

The proposed system is implemented on baremetal and virtual machine autonomously to obtain the benchmark results.

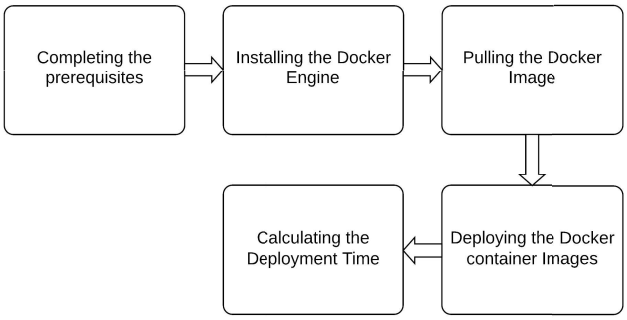

Fig. 2. Steps for executing the benchmark.

Fig. 1. Shows the architecture of the proposed system depicting the layers involved for benchmarking Docker container-based application. This architecture is present in both the scenario of the baremetal and virtual machine. The Docker container application is deployed on the Docker Engine where the startup time is assessed by the Benchmark tool. Initially, both the systems are kept ready by completing the prerequisites of the benchmark. The experimental setup is based on Linux distribution, and the benchmark tool is written for the Bash shell. The benchmark tool is an automated script for deploying Apache web service capable of calculating the startup time for each deployment. The web services are implemented in the count of 20, 30, 40, and 50 to analyze the performance. This count of implementation is based on the hardware on which the benchmark is running.

IV. Methodology

The methodology adopted for evaluating the startup time of Docker container application is simple to implement. Fig. 2 illustrates the steps involved in running the benchmark. These steps were performed on the baremetal and virtual machine to obtain the time for deploying the containerized web server.

V. Docker

Docker is developed in Go Programming Language and uses features of Linux Kernel. Docker system consists of the runtime of the packaging tool, Docker Engine (portable and lightweight), and Docker Hub [13]. For isolation, it uses various isolation groups of Linux such as namespace, cgroups, and UFS [14].

Docker uses namespace for isolating workspace of containers. For that specific container, Docker creates a set of namespaces [15]. Each container runs within that namespace and does not have any access outside it [15]. This isolation makes sure that each process is separated by not allowing them to know the resource held by other groups.

Control groups (cgroups) provide control over resources. Cgroups allows Docker for sharing available resources for giving efficient multi-tenant environment on the host [16]. Cgroups can have reservation or limitation to resources for a container.

UFS (Union File System), or UnionFS is a file system used for making Docker lightweight and faster by creating layers. Docker uses UnionFS variants such as AUFS, btrfs, vfs and Device Mapper.

Container format combines all these components and packages it. Default format 'libcontainer' is used for container wrapping. Docker is expected to support other container formats such as BSD Jails or Solaris Zones.

Docker images are created to provide service of a particular application. To share mutual files, increase image download speed and minimize disk usage, Docker containers use the layered file system. For providing storage and sharing container images, DockerHub is used by using two types of repositories. The private repository is used for making the images accessible within a particular organization. Public repository shares the access of repository by giving open access. Docker container can run on all popular Linux distros due to its open standards. Docker containers can run on Virtual Machine, Native Hardware, cloud infrastructure provided by Red Hat, Google, Microsoft, OpenStack, Amazon and other cloud providers [17] [18] [19]. Every Docker container will have its root file system, processes, memory, and network. Docker containers are configured to open ports for communicating with outside world and can map volumes to directories on the host. Likewise, Docker containers can also be connected to having communication between them.

VI. Benchmark

By looking at system specifications, it is difficult to compare the performance of systems. There are different types of benchmark: Real program, Component Benchmark / Microbenchmark, Kernel, Synthetic Benchmark, I/O benchmarks, Database benchmarks and Parallel benchmark [20].

A benchmark is a test which is used to measure the performance of computer, software, or hardware. These tests give an idea about the platform regarding performance. To achieve a particular advantage from the system, we implement the benchmark. Benchmarking is a slow and complicated process tool.

Low-level Benchmarks are used for measuring the performance of the components like CPU clock, memory cycles. High-level Benchmarks are used for measuring the performance of the hardware drivers and operating system. The efficiency of the container is calculated using various benchmarking tools like Lbench, N bench and UNIX bench. Lbench evaluates performance congestion in a wide range of system application and to check systems ability to transfer data between the processor, cache, memory, and disk. Likewise, N bench is used to test the CPU, GPU, and memory evaluated on cache, memory, integer and floating point results. Unix bench focuses on a different aspect of operating system functions like process spawning, inter-process communication(IPC) and file system throughout.

VII. Experimental setup

The environment for the execution of the proposed system is based on two machines, baremetal and virtual machine. The hardware used for experimentation is respective to each device whereas the software is common to both.

The hardware used for implementation of the benchmark on baremetal has Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz, 16GiB RAM, and 4MiB L3 cache. The virtual machine is running on the emulated hardware consisting of 4 CPU cores running on Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz, 16GiB RAM, and 4MiB L3 cache.

The newly installed operating system used is Ubuntu 16.04.4 LTS along with Docker 18.03.1-ce on both of the platforms. The virtual machine used was built on KVM/QEMU 2.5.0. The newly implemented setup will ensure that the overhead in performance by other factors is eliminated. The benchmark tool used is capable of installing the Docker Engine along with the prerequisites. There are two script files used for benchmarking; the first performs benchmark and the other for cleaning the benchmark environment. The results obtained were stored in a text file for further analysis.

VIII. Evaluation

The benchmark is used for evaluating the startup time of Docker container applications on the baremetal and virtual machine. The startup time is the time taken for launching an app in a Docker container. In our experimentation, we have ignored the network latency, disk I/O, and CPU speed. The image used for starting a Docker container is the httpd2.4 standard image with the minimal hosting of web pages. The purpose of using a standard image of httpd is to maintain a standard benchmark environment. During the evaluation, we are deploying multiple web server applications on both the systems. The startup time is calculated for the baremetal and virtual machine by taking the average time of the applications. The results are iterated ten times for each case. Table I, II, III, IV shows the results obtained by deploying web server container images for 20, 30, 40, 50 applications respectively.

Fig. 3. shows the comparison between baremetal and virtual machine obtained by using the benchmark. It shows that the virtual machine is approximately 50% slower than a baremetal machine for starting a Docker container. The startup time of the virtual machine is delayed due to the presence of emulated hardware. This incurs overhead in the performance of the virtual machine. The performance for starting Docker container images is faster than a virtual machine. The results show that the count is not affecting the performance on starting the Docker containers. Some specific processes are scheduled while executing the benchmark which causes a slight variation in performance.

	Baremetal	**Virtual Machine**
Total time for ten results	173.003806869	298.306782901
Average time	17.3003806869	29.8306782901

TABLE I
Result for deploying 20 Docker images.

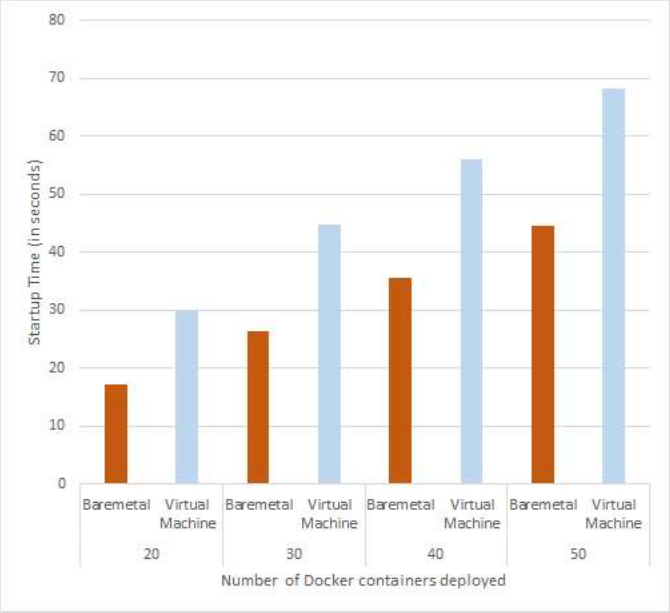

Fig. 3. Comparison results of Baremetal and Virtual Machine.

	Baremetal	**Virtual Machine**
Total time for ten results	263.494891119	447.09369812658
Average time	26.3494891119	44.709369812658

TABLE II
RESULT FOR DEPLOYING 30 DOCKER IMAGES.

IX. CONCLUSION AND FUTURE WORK

This paper gives a clear overview of the need for deploying Docker containers in the baremetal environment. The results show that virtual machine is slower than baremetal which effects the performance in starting the Docker containers. Docker containers outperform in baremetal over the virtual machine by approximately 50%. This degradation in performance of the virtual machine is due to its architecture. The virtual machine runs on emulated hardware thus introducing additional layers than the baremetal. The lack of performance in a virtual machine is not experienced when working with a smaller count of Docker containers, whereas if the number is increased the performance decreases. To utilize the benefits of the Docker containers to the fullest, they must be deployed on baremetal machines.

The further enhancement of the work includes developing the benchmark to consider all the scenarios affecting the benchmark results. The benchmark needs to be performed on high-end hardware which will help in deciding the deployment platform of Docker containers.

ACKNOWLEDGMENT

The work was supported by the Centre for Development of Advanced Computing, Pune, India.

	Baremetal	**Virtual Machine**
Total time for ten results	356.316286605	559.6857347996
Average time	35.6316286605	55.96857347996

TABLE III
RESULT FOR DEPLOYING 40 DOCKER IMAGES.

	Baremetal	**Virtual Machine**
Total time for ten results	445.878225605	681.284650156
Average time	44.5878225605	68.1284650156

TABLE IV
RESULT FOR DEPLOYING 50 DOCKER IMAGES.

REFERENCES

[1] E. N. Preeth, F. J. P. Mulerickal, B. Paul, and Y. Sastri, "Evaluation of docker containers based on hardware utilization," in *2015 International Conference on Control Communication & Computing India (ICCC)*. IEEE, nov 2015.

[2] Z. Kozhirbayev and R. O. Sinnott, "A performance comparison of container-based technologies for the cloud," *Future Generation Computer Systems*, vol. 68, pp. 175–182, mar 2017.

[3] K. Cacciatore, P. Czarkowski, S. Dake, J. Garbutt, B. Hemphill, J. Jainschigg, A. Moruga, A. Otto, C. Peters, and B. E. Whitaker, "Exploring opportunities: Containers and openstack," *OpenStack White Paper*, vol. 19, 2015.

[4] A. Tosatto, P. Ruiu, and A. Attanasio, "Container-based orchestration in cloud: State of the art and challenges," in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*. IEEE, jul 2015.

[5] T. Harter, B. Salmon, R. Liu, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Slacker: Fast distribution with lazy docker containers," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*. Santa Clara, CA: USENIX Association, 2016, pp. 181–195. [Online]. Available: https://www.usenix.org/conference/fast16/technical-sessions/presentation/harter

[6] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, mar 2015.

[7] J. Shetty, S. Upadhaya, H. Rajarajeshwari, G. Shobha, and J. Chandra, "An empirical performance evaluation of docker container, openstack virtual machine and bare metal server," *Indonesian Journal of Electrical Engineering and Computer Science*, 2017.

[8] A. Lingayat, R. R. Badre, and A. K. Gupta, "Integration of linux containers in openstack: An introspection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 3, Dec. 2018.

[9] M Ali Babar and B. Ramsey, "Evaluating docker for secure and scalable private cloud with container technologies," CREST, University of Adelaide, Adelaide, Tech. Rep., 2017.

[10] Y. Yamato, "OpenStack hypervisor, container and baremetal servers performance comparison," *IEICE Communications Express*, vol. 4, no. 7, pp. 228–232, 2015.

[11] A. Lingayat, A. Singh, V. Naik, R. R. Badre, and A. K. Gupta, "Horizon, a web-based user interface for managing services in openstack: An introspection," in *9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2018, pp. 942–945.

[12] M. FISHER, "Performance benchmarking physical and virtual linux environments," mathesis, UNIVERSITY OF CAPE TOWN, 2012.

[13] A. Calinciuc, C. C. Spoiala, C. O. Turcu, and C. Filote, "OpenStack and docker: Building a high-performance IaaS platform for interactive social media applications," in *2016 International Conference on Development and Application Systems (DAS)*. IEEE, may 2016.

[14] V. RATAN, "Docker: A favourite in the devops world," Feb. 2017. [Online]. Available: http://opensourceforu.com/2017/02/docker-favourite-devops-world/

[15] D. Liu and L. Zhao, "The research and implementation of cloud computing platform based on docker," in *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP)*. IEEE, dec 2014.

[16] P. Menage, "Cgroups," Mar. 2018. [Online]. Available: https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

[17] C. Pahl, "Containerization and the PaaS cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, may 2015.

[18] R. Peinl, F. Holzschuher, and F. Pfitzer, "Docker cluster management for the cloud - survey results and own solution," *Journal of Grid Computing*, vol. 14, no. 2, pp. 265–282, apr 2016.

[19] A. Lingayat, V. Naik, A. Singh, S. Wankhade, and N. Dhobale, "User interface for managing openstack magnum service using openstack horizon," in *Two Days 2nd National Level Conference on Emerging Trends in Computer Engineering and Technology (NCETCET18)*, Jan. 2018, pp. 386–390.

[20] D. Elmuti and Y. Kathawala, "An overview of benchmarking process: a tool for continuous improvement and competitive advantage," *Benchmarking for Quality Management & Technology*, vol. 4, no. 4, pp. 229–243, 1997.