Docker in Online Education: Have the Near-native Performance of CPU, Memory and Network?

Basheer Riskhan and Raza Muhammad

***Abstract*—Advent of the cloud computing revolution, Docker technology becomes the crucial part of the virtualization in terms of high resource utilization, less overheads and better performance. Docker is evaluated by using Online Education system based on various extended level of users and extended level of Dockers by using Physical Machine and Linux benchmarking tools. CPU, Memory and Network resources are used as experimental variables. Finally a details analyst compare among the user level and Docker level. As a result, we conclude Docker has the near native performance during the usage of OE system, archived high resource utilization in single Docker by accessing more user as compare to more Dockers by accessing more users.**

***Index Terms*—Docker, performance, online education system.**

I. Introduction

Container based virtualization (CBV) and Hypervisor based virtualization are the most popular virtualization technology at current trend. We noticed the vast differentness in the performance of OE system during the extension of the Virtual Machine (VM) and User Level (UL) while using the hypervisor in my first research. Due to that we will intent to implement Docker to the Online Education system (OES) and verify how the performances deviate from native to Docker during the extension levels.

Recently containers implemented more project and development instead of hypervisor. A Docker is not fully virtualized systems which abstract the OS kernel. More Docker definition mention, that Docker has given the native performance as compare with Physical Machine (PM). So we plan to test, whether the Docker definitions become true or false during the execution of the OES. Docker registry has contained the images and application in cloud which is maintaining by Docker Company. If we need any images or applications, we can grab and containerize We divided three sections as CPU, Memory and Network which are more appropriate host resources for finding the performance of OES. The Linux command, Perl scripting and OES has been used as an experimental tools. Overall Execution percentage time, Resource Utilization, delay time of data sending and receiving are the element we used to measure the matrices in this experiment.

Manuscript received March 24, 2016; revised January 13, 2017.

This work was supported by the Department of Computer Science, Huazhong University of Science and Technology – Docker in Online Education: Have the near-native performance of CPU, Memory and Network.

Basheer Riskhan and Raza Muhammad are with the Department of Computer Science, Huazhong University of Science and Technology, Wuhan, China (e-mail: riskhan@yahoo.com, razacom_2000@yahoo.com).

II. Related work

The author Miguel G [1] expecting, the Container Based Technology (CBT) is given the near native performance from the PM resources. Prove above task, conducted the experiment in different kind of CBT using the NAS parallel benchmark. CBT was compared with Xen in terms of performance and isolation. The isolation performance test was conducted separately by using two guest VMs in the same host and resources divided in same manner. Author concludes Xen obtain bad performance in all virtualization. In container, Poor isolation and less security were observed in resource management implementation. But in the Isolation, Xen has shown the better performance due to non shared OS.

III. Docker Technology

Docker is one of the CBT released on March 2013. This is offer the packaged and deployed the applications, compact inside the virtual container, runs across the all Linux distributions [2]. It has their own file system, libraries, network, etc [3]. Name spaces and Cgroup are the two main features in the Docker. Each Docker execute it's their own environment, not affect or not executing the processes inside of the other Docker. Restricted the file system such as chroot and provide the illusion by the way of wrapped the global resources into the namespace layers are the responsibilities of the namespaces [4]. CPU, I/O usage, memory and network can be carried by Cgroup. Portable Deployment, Security, Isolation, Resource Sharing and allocation, Version Control, Less Time are some benefits of Dockers.

IV. Experimental Setup, Methodology

The experimental setup for compare the performance of OES fully depended on PM and its container. We plan to conduct this experiment on real large scale system during the execution of OES in some stages such as PM execution, PM with containers execution.

A. Physical Machine and Container Setup

PM, networking and storage are the three main sections having the significant impact on the performance of the PM. Docker has running on top of this special hardware layers. The performance of OES is validating through the performance of PM. The hardware specification of PM shows on Table I.

DOI: 10.7763/IJCTE.2017.V9.1154

TABLE I: PM SPECIFICATION

Processor	Intel Core 4200 series 2.4GHz
Main Stream	Dell power edge R710
Memory	8 GB
Hard Disk	1024 GB SATA
Cache Memory	1642 MB
Host OS	Ubuntu server 12.04.3 LTS
Application	OE system
Web Server	Nginx
File System	Ext 3

We developed the real time OES which was our target application installed in the PM. The docker installed on top of PM. Then grab the image from docker registry (# docker pull <OS image name>) and installed in the container (# docker run it <image name> /bin/bash). Then installed nginx web server inside the same container (apt-get install nginx) and did some configuration. Continually connected to the host directory to container directory (docker run -d -p -v <host directory> <container directory> <image name> <container name>). Next we set the network port (docker run -d -p <local port > < container port> <image name>) for outside user access.

B. Experimental Variable and Tools

This experiment evaluates the performance of the PM in terms of CPU, Memory and Network resources. In CPU, we measured the variable as CPU Usage, I/O wait time and idle time [5]. The CPU Usage described to show how much the processor is working currently used to do operations. I/O wait is the time during which that CPU was idle and that has at least one I/O in progress requested by task scheduled on that CPU [6]. Unused or idle percentage of CPU called as idle time. The percentage of using memory includes buffer memory and cache memory out of total memory called as Memory Usage (MU) [7]. The kind of delay occurs in the round trip over the network in data communication called as Network Latency (NL). The entire variable measured in PM from percentage except NL. NL measured from client side which has also indirectly depends on the performance of the OES and measured by micro seconds [8].

As an experimental tool, we used Linux commands with Perl scripting. The OES developed using HTML with CSS, PHP, MySql and java script. The analysis has done by using OrginPro8 software. We used SAR (SAR –u <interval> < total time>) command which is called as System Activity Reporter. It has a command line library of Ubuntu and provides the hardware performance counter in the processor. We can find the CPU usage by adding “%user, %system”, I/O wait time and Idle time by using SAR. MU measured by using FREE command with Perl scripting in Linux (FREE -m -c <total time> -s <interval> | perl -pe 'print localtime()."" , Which helps to display the current time including their data. NL tested from client machine by using Special Ping command (“# ping IP address | while read pong; do echo “$(date): $pong”;done”) due to display the current time.

C. Methodology

Experiment will be conducted in two ways. In first way, the PM for a single user till 5 users and PM with One Docker for single user till 5 users. The second way, PM with one Docker is for one user. But this experiment one extra Docker added to PM for every user till 5 users. . That means 5 users accessing the OES in 5 Docker. The developed OES store in the PM and it mapped with Docker. OES executed, measured each variable matrix for 150 seconds and having the interval of 3 seconds.

First Way

C1-1U: PM with OES, Access by 1 user
C1-2U: PM with OES, Access by 2 users
C1-3U: PM with OES, Access by 3 users
C1-4U: PM with OES, Access by 4 users
C1-5U: PM with OES, Access by 5 users
C2-1U: PM and One Docker, Access by 1 user
C2-2U: PM and One Docker, Access by 2 user
C2-3U: PM and One Docker, Access by 3 user
C2-4U: PM and One Docker, Access by 4 user
C2-5U: PM and One Docker, Access by 5 user

Second way

C1 - PM with OES, Access by 1 user
C2 - PM and 1 Docker, Access by 1 user
C3 - PM and 2 Docker, Access by 2 users
C4 - PM and 3 Docker, Access by 3 users
C5 - PM and 4 Docker, Access by 4 users
C6 - PM and 5 Docker, Access by 5 users

Our main idea is to highlights, the native performance of PM that how deviating while increasing the docker during the access of OES.

V. EXPERIMENT AND EVALUATION

This section is discussing the performance of OES through the measured matrix of each variable individually. The first way, result was obtained by increasing the user one by one till five. Similarly this procedure was followed for the PM with one docker. The second way, the level of Docker has increase and each Docker access by each user.

A. CPU Usage

CPU usage measured with the combination %user and %System in every 3 seconds for 50 times during the execution of OES and reported the metrics shows on Table II.

TABLE II: CPU USAGE

Condition		1 U	2 U	3 U	4 U	5 U
C1	PM	1.01	1.03	1.03	1.06	1.07
C2	PM,1D	1.11	1.14	1.16	1.17	1.20
C3	PM,2D		1.17			
C4	PM,3D			1.24		
C5	PM,4D				1.29	
C6	PM,5D					1.33

In first way, CPU usage matrix has gradually increased in C1 and C2. The Fig. 1 shows the result of second way of experiment.

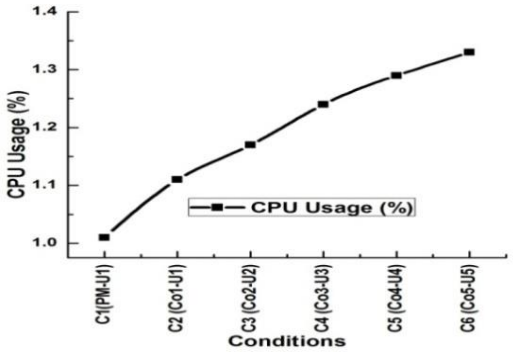

Fig. 1. CPU usage in each docker.

In both ways, found the gradual increment of CPU usage. This may be the cause of processing multitasking, execute more application and need resources for schedule the processing to the Docker.

Further the first way of increment level is lower than the second way. That's clearly said, access the more users to less Docker is better than the more users to more Docker.

B. I/O wait Time

I/O wait time indicates the percentage of CPU cycle waiting for I/O events. Table III shows the reported matrices. The results of I/O wait time increased in C1 and C2 same as CPU usage in first way. The Fig. 2 shows the result of second way.

TABLE III: I/O WAIT TIME

Condition		1 U	2 U	3 U	4 U	5 U
C1	PM	0.27	0.28	0.30	0.31	0.35
C2	PM,1D	0.30	0.32	0.35	0.36	0.40
C3	PM,2D		0.34			
C4	PM,3D			0.37		
C5	PM,4D				0.39	
C6	PM,5D					0.42

Gradual increment observed in I/O wait time like CPU Usage in both ways. Use of web server, Network Issues, OES access by multi users, delay process are might be the causes of increase level of I/O wait time. But these increments have not more harmful to the performance of the OES. This result also motivated to use more users in fewer Dockers.

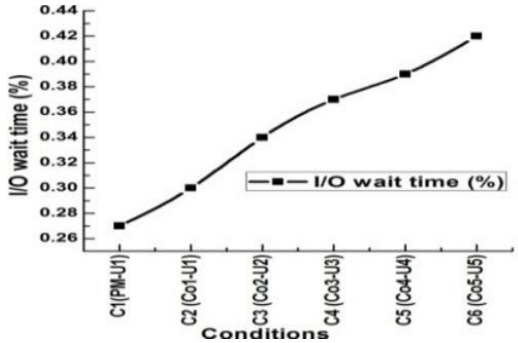

Fig. 2. I/O wait time in each docker.

C. Idle Time

CPU was idle and System did not have an outstanding I/O request called Idle time. Table IV shows the reported matrices.

TABLE IV: IDLE TIME

Condition		1 U	2 U	3 U	4 U	5 U
C1	PM	98.98	98.95	98.84	98.67	98.56
C2	PM,1D	98.89	98.91	98.80	98.60	98.52
C3	PM,2D		98.83			
C4	PM,3D			98.71		
C5	PM,4D				98.61	
C6	PM,5D					98.47

The results of idle time decreased in C1 and C2 in first way.

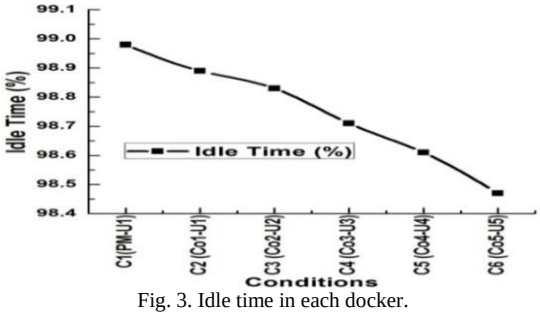

Fig. 3. Idle time in each docker.

The Fig. 3 shows the result of second way. The idle time has decreasing while we extended the level of Docker and user. It may be cause of lot of disk processing. But according to the matrices we understand that the through put was better during the extended of Docker and users.

D. Memory Utilization (MU)

MU measured the usage of PM includes buffered and cached [7]. We used Perl scripting to display the current time.

TABLE V: MEMORY UTILIZATION

Condition		1 U	2 U	3 U	4 U	5 U
C1	PM	23	25	28	28	30
C2	PM,1D	28	29	31	32	35
C3	PM,2D		32			
C4	PM,3D			33		
C5	PM,4D				35	
C6	PM,5D					38

Table V shows the reported matrices. The results of MU increased in C1 and C2 in first way. The Fig. 4 shows the result of second way.

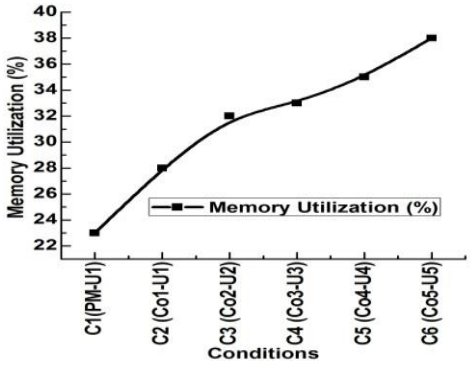

Fig. 4. Memory Utilization in each docker.

Due number context switching which required to fetch the data form main memory is taken high utilization as compare to fetching the same data from cache. Further during the extending the process on the Docker and extending the number of Docker lead to increase the overhead of context switching. These Might be the reason of high utilization of memory.

E. Network Latency (NL)

NL can be varies from application to application. NL measured from client side and special PING test command used for displaying the system time which was the unique element.

TABLE VI: NETWORK LATENCY

Condition		1 U	2 U	3 U	4 U	5 U
C1	PM	104	111/ 109	116/ 124/ 118	117/124/ 129/122	138/122/118/ 132/120
C2	PM,1D	114	118/ 114	130/ 118/ 121	138/134/ 121/126	126/144/128/ 138/136
C3	PM,2D		117/ 115			
C4	PM,3D			117/ 124/ 124		
C5	PM,4D				131/122/ 130/122	
C6	PM,5D					120/138/141/ 126/132

Table VI shows the reported matrices of NL. The delay time increased in the experiment of first way and second way according to the measured results. The Fig. 5 shows the result of second way.

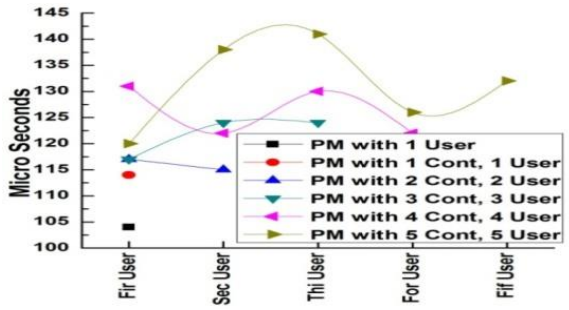

Fig. 5. Network Latency in each docker.

The flow of NL has increase during the extended of Docker level and UL. But the increase level was little low in the first way as compare to second way. Each gateway creating the traffic is one of the causes of High NL.

VI. CONCLUSION

This paper mainly focuses the significant performance of the OES in the Docker. The experiment conducted in two ways which are extended the UL and extended the Docker level. PM resources and the processing time take by processor are the main basic measurement aid in this research. We had compare the three stage which are real execution of PM and its extended UL, real execution of PM with One docker and extended UL, real execution of PM with extended level of Docker. Through that found the three innovative concepts. 1. The performance impact observed to the OES between native execution and Docker based execution, but it is near-native performance. 2. The measures matrixes have not varied constantly it's varied in terms of the usage of Docker and the usage of OES. 3. Performance degradation of OES is higher in the extended level of Docker as compare to extended level of user in single Docker. Further in the both ways noticed the minute performance loss in the all variables during the extended level. But as compare to the benefits of Docker, this performance loss is nothing. Through the experiment, we recommended to implement Docker technology into OES and allow access the OES by more users in single Docker.

REFERENCES

[1] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. F. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," *2013 21st Euromicro International Conference on Parallel Distributed and Netowrk-Based Processing*, Belfast, 2013, IEEE, pp. 233-240.

[2] X. Skin. (July 2014). New linux container virtualization technology from docker. *HowToForge.* [Online]. Available: https://www.howtoforge. com/docker-linux-container-virtualization.

[3] , A. Ghosh. (July 2014). What is container based virtualization? *Computer and Internet.* [Online]. Available: https://thecustomizewindows. com/2014/07/container-based-virtualization/.

[4] A. M. Joy, "Performance comparison between Linux containers and virtual machines," *2015 International Conference on Advances in Computer Engineering and Applications (ICACEA)*, Ghaziabad, 2015, pp. 342-346.

[5] I. Paul, S. Yalamanchili, and L. K. John, Performance impact of virtual machine placement in a datacenter, *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*, Austin, TX, 2012, pp. 424-431.

[6] Linux. (Nov. 2013). The precise meaning of I/O wait time in Linux. *Veithen.Github.io.* [Online]. Available: http://veithen.github.io/2013/11/18/ iowait-linux.html.

[7] S. Anne. (May 2013). Understanding free command in Linux/Unix. *The Linux Juggernaut.* [Online] Available: http://www.linuxnix.com /find-ram-size-in-linuxunix

[8] A. Tikotekar, H. Ong, S. Alam, G. Vall é,T. Naughton, C. Engelmann, and S. L. Scott, "Performance comparison of two virtual machine scenarios using an hpc application", *2009. 3rd Workshop on System-level Virtualization for High Performance Computing*, ACM, pp. 33-40.

Basheer Riskhan was born on July 13th, 1977 in Jaffna, Sri Lanka. He received primary and secondary education at St. John's College, Jaffna and Zahira National College, Puttalam. He earned his bachelor degree in computer science from Bharathidasan University, India in 2002 and master degree in education from National Institute of Education, Sri Lanka in 2012. At present he is pursuing his Ph.D in computer science from Huazhong University of Science and Technology Wuhan, P.R. China.

Basheer Riskhan joined National College of Education in Sri Lanka as a Lecture in 2005 and working as a Sri Lanka Teacher Educator Service (SLTES) officer in Sri Lanka. Before that he served as Software Engineer in several private companies. His research interests are in the field of Virtualization, Cloud Computing, Kernel Programming and Big Data. He had published several research papers and participated in various conferences as an oral Presenter. His main concern is integrating education and information technology together.

Raza Muhammad was born in 1981 in Kharpur, Sindh, Pakistan. He completed his bachelor degree BS(CS) for Shah Abdul Latif University Kharipur Pakistan in computer science in 2004 and move to Karachi for higher studies. In 2007 he completed his master degree MS (CS) in computer science from Pakistan Air Force-Karachi Institute of Economics and Technology Karachi Pakistan. Now he is pursuing his PhD in computer science from Huazhong University of Science and Technology Wuhan P.R. China.

In his academic career he worked as lecturer and visiting lecturer in various universities and government institutes. He also contributed in research papers as main author and as second author. His area of interests are big data, Linux Kernel programming, wireless network and security system of computation.