ASEE's 123rd Annual • Conference & Exposition • New Orleans, LA • June 26-29, 2016

New Orleans JAZZED about Engineering Education

Paper ID #16188

# Adoption of Container-Based Virtualization in IT Education

**Dr. Tae-Hoon Kim, Purdue University - Calumet**
**Dr. Keyuan Jiang, Purdue University - Calumet**
**Mr. Vivek Singh Rajput**

Adoption of Container-based Virtualization in IT Education

## Abstract

Hands-on experience through the lab is one of the key components in Information Technology education because it provides students with an opportunity to learn and observe how to apply the concepts. Generally, the lab in IT education requires a variety of equipment such as PCs, servers, switches, and so forth. And virtual environment becomes an essential component in IT education since virtually multiple systems can be created, modified, tested, and deleted easily with little or no cost. Hypervisor virtual machines have been heavily used in IT education for last decade, but sharing the large resources with the host machine becomes a major limitation of such technology. Recently, emergence of container-based virtual machines provides a lightweight virtual environment over hypervisor-based machines. This emerging technology has not been explored well and adopted in IT education. In this paper, we will investigate and explore container-based virtual machines to provide framework for its adoption in IT education. We will describe step-by-step instructions on creating a cluster in single host and across different hosts using container-based virtualization technology. Afterward, we will evaluate and compare the performance of such implementation with a cluster built on physical machine.

## 1 Introduction

The practical exercises included in laboratory-based course play an important role in engineering and science educations. Many academic institutes developed the laboratory-based courses to help students to accelerate their learning in different types of laboratories such as real, simulation, or online [1]. Especially in Information Technology education, hands-on exercises through the laboratory became an essential component of the course because it provides students with an opportunity to learn and observe how to apply the concepts. Generally, the lab in IT education requires a variety of equipment such as PCs, servers, switches, and so forth. The variety of equipment usually incurs the cost, which becomes the main restriction to providing the lab with appropriate environment. Due to the cost restriction, virtualization technologies have been widely adopted in many areas. Since the virtual technology provides the isolated virtual environment from host or other virtual machines, it became very popular for development and education solution. Virtualization technology becomes an essential component in IT education because virtually isolated multiple systems can be created, modified, tested, and deleted easily with little or no additional cost.

Two types of virtualizations are currently available to use, hypervisor-based and container-based virtualization. Hypervisor-based virtualization implementations such as Xen, VMware, and KVM [2-4] have been heavily and widely used in IT education for last decade. The characteristic of the hypervisor virtualization allows allocating the hardware resource of host machine to create the isolated virtual environment, which has its own Operating System (OS). Resource abstraction becomes the major limitation of such technology because it degrades the system performance typically. Due to the high performance overhead, traditional hypervisor-based virtualization has not been considered in some place including High Performance Computing (HPC) [5-7].

Recently, container-based virtualization technology has emerged, which provides a lightweight virtual environment over hypervisor-based machines. This emerging technology uses the container concept to create the isolated virtual environment. Instead of hardware resource sharing, it creates a virtual environment on the top of Linux OS kernel. The minimal use of resource makes this type of virtualization much lighter than traditional hypervisor-based virtualization with similar performance. The container-based virtualization technology has not been explored well, nor adopted in IT education. A few literatures studied the container-based virtualization in its performance and adaptation [7-9]. In this paper, we will investigate and explore the container-based virtual machines to provide framework for its adoption in IT education, especially in the HPC course. We will implement both virtualization tools to build the Beowulf cluster and describe the instructions on creating a cluster in single host and across the multiple hosts using both virtualization technologies. Afterward, we will also evaluate and compare the performance of such implementations with a cluster built on physical machine.

The rest of paper is organized as follows. Section 2 describes the computer cluster including Beowulf cluster and its building process. Section 3 describes and compares both virtualization technologies. The experimental setup is described in Section 4 and the results are illustrated and discussed in Section 5. Finally, Section 6 concludes the paper.

## 2 Computer cluster

HPC is heavily used in computational science these days to process the large volume of data such as daily stock analysis, weather forecasting with simulation, animation, and etc. Supercomputer consisted of a massive number of multi-core processors was developed for high-level computing capacity. Computer cluster became one of the approaches to build the supercomputer, as the commodity high performance processors, low latency network, and software tools to use the cluster were available. Individual processors used in personal computers became highly powerful and they can be connected through a low latency network to create the cluster. To facilitate the use of cluster, Message Passing Interface (MPI) is commonly implemented. MPI is a standard portable programming interface for parallel computing, which supports point-to-point and collective communication [10]. The abstractions in two levels of processes, such as process rank and virtual topology, are provided by MPI. It also provides a set of library functions and supports single program, multiple data (SPMD) and multiple programs, and multiple data streams (MPMD).

Beowulf cluster is a high performance computer cluster by use of less powerful computers such as personal computer. It was developed and built in 1994 by Thomas Sterling and Donald Becker at NASA using less powerful PCs for high performance scientific workstations in the earth and space sciences community [11,12]. It uses less powerful commodity hardware, on a private system network, with open source software (i.e., Linux) infrastructure [17]. Due to the cost effectiveness, it has been widely adopted and used in traditional technical applications such as simulations, biotechnology, data mining, stream processing, and etc. In addition, it is also heavily used in academic research and education fields [13-16]. Beowulf cluster has two structural components; therefore, it requires two separate installations on one master node and multiple slave nodes, which should be connected through low latency network such as LAN with static IP address for the massage passing. Three major separate setups are required to build Beowulf cluster; Network File System (NFS), Secure Shell (SSH), and MPI.

## 3 Virtual machines

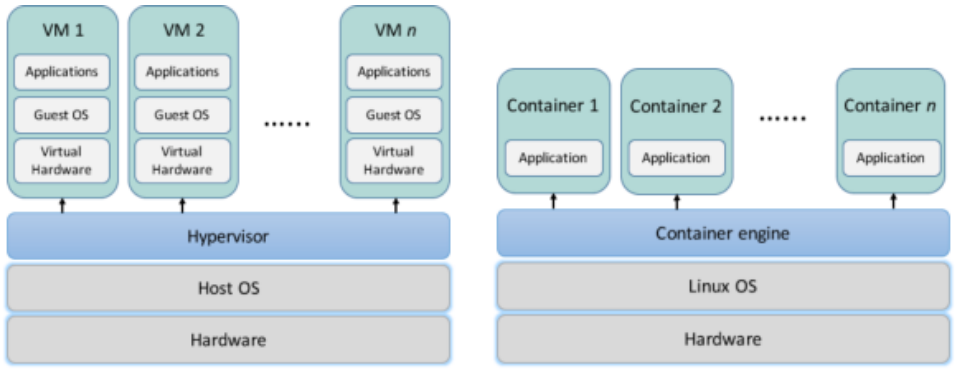
Virtual machine (VM) is a software-implemented emulation of the computer system and executes the program as a physical machine. The purpose of virtualization is to execute the program in different environments from the host machine. Two techniques are developed and available, hardware virtualization and abstract virtual machine. Since the hardware virtualization virtually creates one or more separate physical space environment(s) isolated from the host machine, it is preferred and widely adopted. In this paper, we will focus on and compare these two virtualization approaches.

### *3.1 Hypervisor-based virtual machine*

Hypervisor-based virtual machine is a software and/or hardware implementation to create and run separate virtual environments different from host machine. The type-1 hypervisor, called native or bare-metal hypervisor, controls the host machine hardware to manage the guest OSs. Oracle VM Server for SPARC, Citrix XenServer, VMware ESX/ESXi, and Microsoft Hyper-V are the type-1 hypervisors. The other type is the hosted hypervisor or type-2. Type-2 hypervisors creates the guest OS from host OS. VMware workstation, VMware player, VirtualBox, QEMU are the examples of type-2 hypervisors. Both types abstract the guest OS in software and/or hardware. Hypervisor is running on the top of the host machine's OS and allocates the host machine hardware resources, such as file system, storage space, and memory, for each guest OS as shown in Figure 1(a). Due to the hardware resource allocation, creating and running guest machine is resource-heavy. The number of simultaneously running guest machines is greatly dependent upon the host machine specification. Therefore, the higher end system specification is preferred in hypervisor-based virtual machine. However, it is beneficial to run the application in different types of OS since it allows creating and running different type of OS in each guest machine. Each guest machine has its own resources and OS in hypervisor-based virtual machine.

### *3.2 Container-based virtual machine*

Unlike hypervisor-based virtual machine running a full OS, container-based is running on the top of the existing OS as shown in Figure 1(b), which provides the isolation using a kernel namespace feature in Linux. The basic idea of container-based virtualization concept is to minimize the resource usage by sharing the underlying kernel, but the processes inside the container are still running on the Linux system. Since the container abstracts the OS kernel rather than the hardware, it could be a lot lighter as small as a single process. It could be either system or application container. The former runs init, inetd, sshd, and etc. while the latter runs an application and does not consume the memory [19]. The consumption of the resources such as memory and CPU is managed by the control subsystem in Linux. Some resources can be shared among containers such as folder and file sharing by use of bind mount. It also provides efficient communication between containers or between a container and the host machine.



(a) Hypervisor-based (b) Container-based

**Figure 1.** Structural comparison of hypervisor-based and container-based virtualization

Linux containers are managed by the management tool and several of them are available, such as LXC, system-nspawn, lmctfy, Warden, and Docker [20-24]. The commonly known Linux container by people is LXC, which is a set of tools, templates, library, and language bindings [18]. LXC creates a container as a virtual machine with a fully functional OS and therefore multiple applications can be running in a single container. Recently, another management tool, Docker, has been showing rapid growth and attracting many attentions from education, research, and industry due to its features and ease of use. Docker uses AUFS (Another UnionFS) to provide layered stack of filesystems, which reduces and simplifies the space and management of filesystem [25]. Since it uses overlaying technique to provide the system environment for individual container based on a single OS image, it typically consumes relatively less resources in disk space and I/O.

## 4 Experimental setup

In this session, we compare the hypervisor-based and container-based virtual machine using VirutalBox and Docker, which are freely available. VirutalBox is general-purpose virtualization tool developed by ORACLE and compatibly runs on many types of OS, Windows, Mac OS X, Linux, and etc. [26]. VirtualBox allows user to select the guest OS resource allocation but it requires minimum memory and disk space. For each guest OS, minimum of 512 MB memory and 8 GB of hard disk space, but larger memory is recommended and hard disk space will easily grow to several dozen GBs. A container-based virtualization (i.e., Docker) is running on the top of a virtual Linux OS, which uses the VirtualBox to run Linux OS for non-Linux type OS. A container does not require memory or large hard disk space. It requires several hundred MBs of hard disk depending on the container setup.

We compare these two virtualizations in two scenarios. The first scenario is to create a Beowulf cluster in one host and another in multiple hosts.

### *4.1 Networking in virtual machines*

Beowulf cluster implements the MPI, which requires the communication between master and slave nodes through LAN in general. The networking is an essential to build the Beowulf cluster. Both virtualizations provide with networking capability in both internal and external networks.

VirtualBox provides three essential networking types, Network Address Translation (NAT), Bridged networking, and Host-only networking. Initially, VirtualBox creates a virtual LAN internally to provide the communication between host and guest VMs and among guest VMs. The host-only networking allows internal communication only and guest VMs cannot see external devices and vise versa. Both NAT and Bridged networking enable guest VMs to communicate with external devices. The difference between NAT and Bridged networking is that the external machine only sees the host in NAT while guest VMs can be seen in Bridged networking.

The networking in Docker has not been well developed yet compared to hypervisor-based virtualization. The default network in Docker is bridge network named bridge appearing as docker0, created at the installation. Unlike VirtualBox, the bridge network in Docker does not allows external machines to see the containers in the Docker, but internal containers in the single host can communicate each other through the bridged network. For the service to the external network through the container, it uses port mapping and/or forwarding. One of any well known and unknown ports in the host machine can be allocated and mapped to the specific container desired. For example, Docker can map the port 80 to the container specifically running a web server and it can be seen and accessed by external devices.

### *4.2 Beowulf cluster through virtual machine*

Beowulf cluster requires two or more nodes and its building process includes three major setups, NFS, SSH, and MPI. We use two PCs with different specifications. One PC has quad-core Intel® Core™ i7-3610QM CPU @ 2.30 GHz with 8 GB of RAM. The other has dual-core Intel® Core™ i5 CPU 470 @ 1.33 GHz with 4 GB of RAM. One of Linux OS, Ubuntu, is installed on both PC. We allocate 512 MB of memory for each guest VM in VirtualBox.

#### *4.2.1 Single host*

We use the quad-core PC to test the cluster in a single host. In both virtualizations, default networking enables the communication among all internal nodes, master and slaves. In VirutalBox, all three networking types, NAT, Bridged, and Host-only, provide internal communication among guest VMs. Default networking in Docker is bridged networking, which allows the communication among internal containers. IP address should be assigned to all guest VMs or containers and static IP is recommended for all participants. In VirtualBox, static IP address can be easily configured at the virtual Ethernet interface created in the guest VM as in Linux OS. Docker, container-based virtualization, can assign static IP address when the container starts with an option. Note that the same network IP address should be assigned to all guest VMs or containers. All other steps, NFS, SSH, and MPI, can be easily configured in VirtualBox and Docker except the NFS in Docker. In Docker, other method (i.e., mirroring) should be used to share the files and folders, which has to be exported first when the master node container starts. The sharing folder mounting in any other containers should be done with mounting option at Docker before the container starts. In the single host test, we examine the cluster performance by the number of creating virtual nodes (i.e., 2, 4, 6, and 8 nodes).

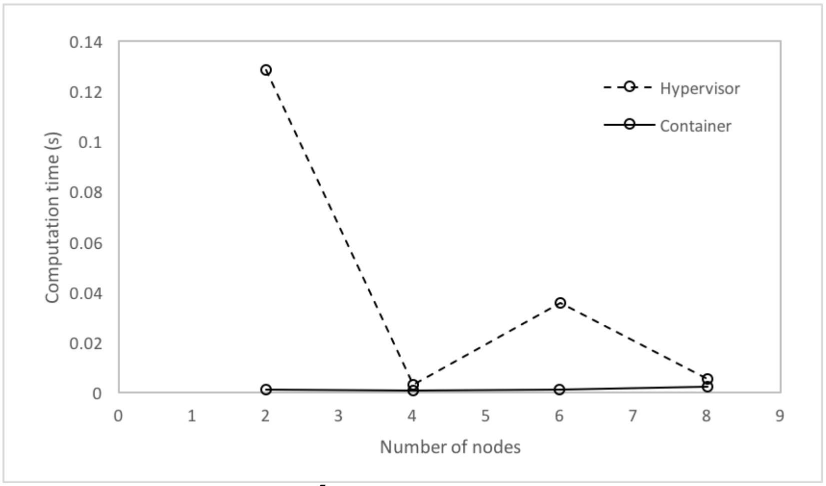
### *4.2.2 Multiple hosts*

We also implement the Beowulf cluster across the multiple hosts. VirtualBox provides the Bridged networking, which enables guest VM to see another guest VMs in other hosts by creating a virtual LAN environment. IP addresses should be configured appropriately to ensure that they are in the same network, which are virtually in the LAN (i.e., same network addresses). Docker supports port forwarding technique for the communication between containers in multiple hosts. Docker binds one or more ports in the host machine to the container and makes them to indirectly see each other through the host. For example, a container can run a SSH server to provide the service by binding the port number 22 to the container's. MPI uses SSH for the communication between master and slave nodes. Port binding and forwarding technique is able to establish the SSH, but further communication for the message passing is not available. It may be due to inconsistency of IP addresses because master and slave containers only know the IP address of corresponding host machine. For example, the master node container knows the IP address of the host machine running the slave node container, not slave node container's IP address. This may cause the problem for the further message passing beyond the SSH connection between master and salve nodes. To overcome such a problem, we use another solution, Open vSwitch, which is an open virtual switch ported to multiple virtualization tools including KVM, VritualBox, and etc. [27]. It also supports Docker to create a virtual LAN environment among containers across the multiple hosts by modifying the interfaces file at "etc/network/" folder. Once all containers are networked in the virtual LAN environment, the cluster setup is similar to the single host. Similarly, all VMs or containers should have the same network address and static IP address assignment is recommended. Mirroring technique should be still used in Docker for multiple host case too. We use two PCs to build the cluster; master in quad-core PC and slave in the other.

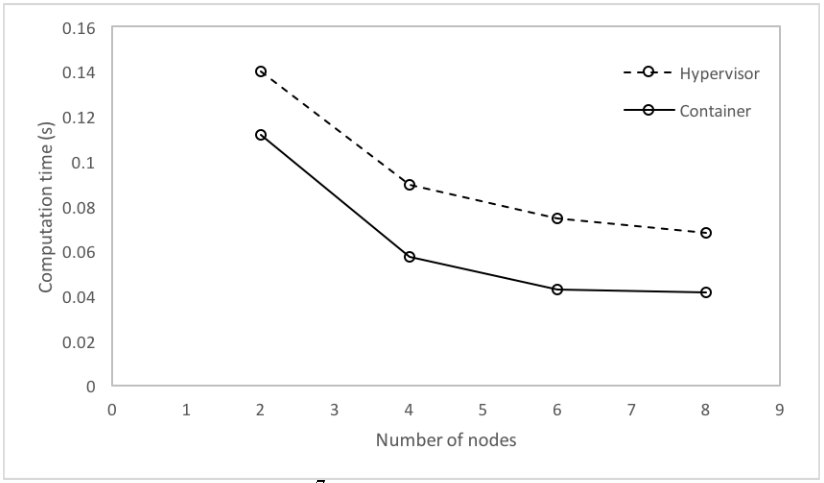
## 5 Results and Discussion

Here we use simple "pi calculation" to compare the clusters built in hypervisor and container. The master node splits the computation jobs uniformly to the available nodes and displays the result. The start and end times are recorded at the master node to measure the total computation time for the comparison. Total computation time includes all necessary components in MPI such as communication, processing, and queueing time. Let us define the variables for the comparison in the rest of this paper. Let $T_{\mathrm{A}}$ be the total computation time in system A. Let $D_{\mathrm{AB}}$ be the difference of computation time that can be computed by subtracting the computation time in system B from that in system A (i.e., $T_{\mathrm{A}} - T_{\mathrm{B}}$). Positive $D_{\mathrm{AB}}$ means computation time in system A is greater than that in system B, which leads us to conclude that system B completes the same amount of computational loads faster than system A.

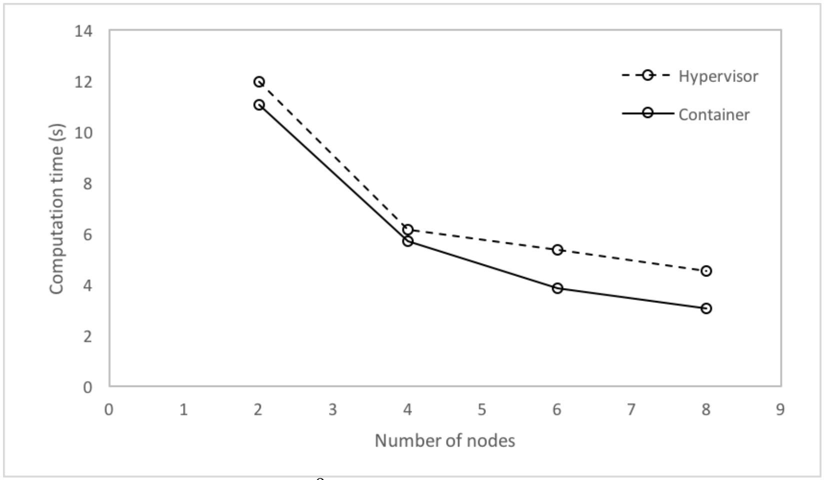
In the single host, we compare the performance of hypervisor with that of container in different number of nodes (i.e., 2, 4, 6, and 8 nodes including master). At each number of nodes, we measure the total computation time (i.e., $T_{\mathrm{h}}$, $T_{\mathrm{c}}$, total computation time in hypervisor and container respectively) for three different numbers of intervals, $10^5$, $10^7$, and $10^9$. Ten measured total computation times are collected and averaged as in Figure 2, which shows the average of total computation time at each number of nodes in (a) $10^5$, (b) $10^7$, and (c) $10^9$ intervals. The test results show that the computation time in hypervisor is relatively greater than that in container for all number of nodes (i.e., $T_h > T_c$). This implies that the cluster in container completes the same amount of computational loads faster than one in hypervisor. The difference of the computation time between hypervisor and container (i.e., $D_{hc} = T_h - T_c$) shows how much faster the cluster in container finishes job than that in hypervisor. Figure 3 illustrates the results of the difference in the total computation time, which shows that the difference becomes more significant when the amount of computational load increases. With the relatively lighter computational load (i.e., $10^5$ and $10^7$), the averages of the difference (i.e., $D_{hc}$) are 0.04176 and 0.02970, respectively. When the computational load is relatively heavier (i.e., $10^9$ intervals), the average of the difference is 1.07707, which is approximately 26 times and 36 times greater than $10^5$ and $10^7$ intervals, respectively. Our results show that the cluster built in Docker performs relatively better than that in hypervisor in single host evaluation. It is also noted that not more than 9 nodes cluster (i.e., 9 VMs) can be created using hypervisor (i.e., VirtualBox) in our system. However, more number of containers can be created in Docker. We tested up to 16 nodes to build the cluster in the Docker and no system degradation was observed.



(a) $10^5$ number of intervals



(b) $10^7$ number of intervals



(c) $10^9$ number of intervals

**Figure 2.** Performance comparison of clusters in single host using pi calculation with different number of intervals (a) $10^5$, (b) $10^7$, and (c) $10^9$.

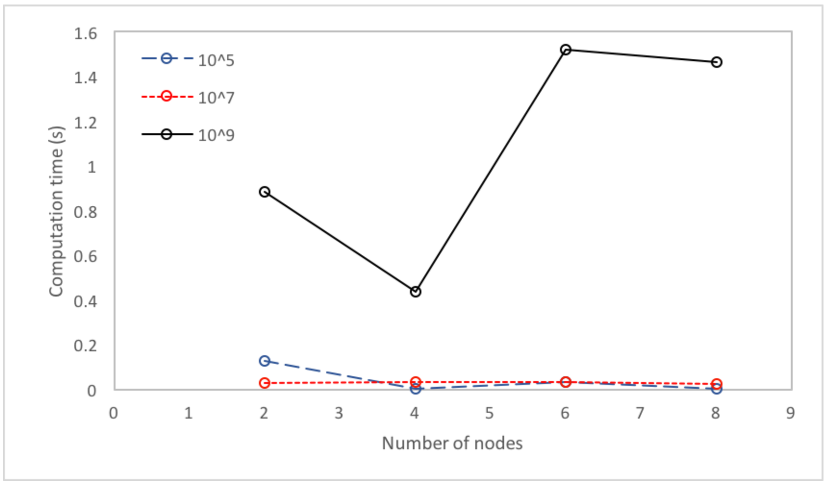


**Figure 3.** Difference in total computation time ($D_{hc} = T_h - T_c$) in different number of intervals.

We use two PCs to create the cluster using both hypervisor-based and container-based (i.e., VirutalBox and Docker). One PC plays as a master and the other as a slave. The results in multiple hosts are illustrated in Table 1. It also shows that the time difference in computation (i.e., $D_{hc} = T_h - T_c$) is not significant at lighter computational loads (i.e., $10^5$ and $10^7$ intervals). But it becomes more significant at higher computational loads (i.e., $10^9$ intervals). One thing to note is that the total computation time in hypervisor at $10^7$ intervals is faster than one in container, which is minimal (i.e., 0.00427). According to the test results, it also shows that cluster built in container-based virtualization (i.e., Docker) performs better, which becomes more significant at the heavier computational load.

**Table 1.** Performance comparison of clusters across the multiple hosts

| Total computation time in second | | | |
|---|---|---|---|
| Number of intervals | $10^5$ | $10^7$ | $10^9$ |
| VirtualBox | 0.16886 | 0.20760 | 22.11854 |
| Docker | 0.01669 | 0.21187 | 20.37149 |

Our study shows that the cluster built in container-based outperforms in both single and multiple host environments. However, the container-based virtualization has the limitations. In container-based virtualization, virtual LAN among containers across the multiple hosts is not supported. Therefore, the cluster across the multiple hosts in container-based virtualization has to use Open vSwitch, which only works with Linux OS. This limits the implementation of the cluster across multiple hosts installed with different operating systems in such a virtualization. In addition, the container-based (i.e., Docker) approach is not easy to manage for those users unfamiliar with Linux OS. Besides these limitations, the container-based virtualization outperforms the hypervisor technology and it can create, modify, and remove the container instantly and easily in contrast to the fact that hypervisor-based virtualization requires a lot more time and resources to load the VM. Overall, the container-based virtualization can be considered as an alternative to or replacement of the hypervisor-based virtualization in IT education.

## 6 Conclusions

Hypervisor-based virtualization has been widely adopted in education to help students to understand course materials. Recently, an emerging technology, container-based virtualization, provides a lightweight virtual environment. In this paper, we introduced the implementation of the Beowulf cluster in both hypervisor-based and container-based virtualizations and compared their performance by total computation time in light and heavy computational loads. Our study shows that container-based virtualization can be used to build the Beowulf cluster in single host and across the multiple hosts. The cluster building process in container-based virtualization is similar to the hypervisor-based one except the networking and NFS. In terms of usability, container-based has some limitations to build the Beowulf cluster such as ease of use and OS restriction due to the networking. However, container is relatively easy and fast to create, load, and remove. In performance evaluation, cluster built in container-based outperforms especially at heavier load in both single and multiple hosts. Our observation shows that the container-based virtualization can be easily adopted in IT education with the similar or even better performance. In the future, we will investigate more in-depth the networking of container-based virtualization to relieve the OS restriction. And we hope that it will help IT education widely adopt the container-based virtualization.

## References

[1] Ma, Jing, and Jeffrey V. Nickerson. "Hands-on, simulated, and remote laboratories: A comparative literature review." *ACM Computing Surveys (CSUR)* 38.3 (2006): 7.
[2] “Xen,” 2012. [Online]. Available: http://www.xen.org
[3] “VMware,” 2012. [Online]. Available: http://www.vmware.com
[4] “KVM,” 2012. [Online]. Available: http://www.linux-kvm.org

[5] Regola, Nathan, and Jean-Christophe Ducom. "Recommendations for virtualization technologies in high performance computing." *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010.
[6] Walters, John Paul, et al. "A comparison of virtualization technologies for HPC." *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*. IEEE, 2008.
[7] Xavier, Miguel G., et al. "Performance evaluation of container-based virtualization for high performance computing environments." *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE, 2013.
[8] Felter, Wes, et al. "An updated performance comparison of virtual machines and linux containers." *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*. IEEE, 2015.
[9] Jiang, Keyuan, and Qunhao Song. "A Preliminary Investigation of Container-Based Virtualization in Information Technology Education." *Proceedings of the 16th Annual Conference on Information Technology Education*. ACM, 2015.
[10] Gropp, William, et al. "A high-performance, portable implementation of the MPI message passing interface standard." *Parallel computing* 22.6 (1996): 789-828.
[11] Becker, Donald J and Sterling, Thomas and Savarese, Daniel and Dorband, John E and Ranawak, Udaya A and Packer, Charles V, "BEOWULF: A parallel workstation for scientific computation", in Proceedings, International Conference on Parallel Processing vol. 95, (1995).
[12] Wikibooks. Building a Beowulf Cluster — Wikibooks, The Free Textbook Project. [online], 2011. http://en.wikibooks.org/w/index.php?title=Building_a_Beowulf_Cluster&oldid=2210594, last viewed December 2014.
[13] Frinkle, Karl, and Mike Morris. "Developing a Hands-On Course Around Building and Testing High Performance Computing Clusters." *Procedia Computer Science* 51 (2015): 1907-1916.
[14] Ngxande, Mkhuseli, and Nyalleng Moorosi. "Development of Beowulf Cluster to Perform Large Datasets Simulations in Educational Institutions." *Development* 99.15 (2014).
[15] Saldaña, Rafael, et al. "Development of a Beowulf-Class High Performance Computing System for Computational Science Applications." *Science Diliman* 13.2 (2012).
[16] Kiepert, Joshua. "Creating a raspberry pi-based beowulf cluster." *Boise State* (2013).
[17] Beowulf Project Overview, www.weowulf.org
[18] LinuxContainers.org Infrastructure for container projects, www.linuxcontainers.org
[19] Felter, Wes, et al. "An updated performance comparison of virtual machines and linux containers." *technology* 28 (2014): 32.
[20] Stphane Graber and others. LXC—Linux containers. https://linuxcontainers.org/.
[21] Lennart Poettering and Kay Sievers and Thorsten Leemhuis. Control centre: The systemd Linux init system. http://www.h-online.com/open/features/Control-Centre-The-systemd-Linux-init-system-1565543.html, May 2012.
[22] Victor Marmol and others. Let me contain that for you: README. https://github.com/google/lmctfy/blob/master/README.md.
[23] Cloud Foundry Warden documentation. http://docs.cloudfoundry.org/concepts/architecture/warden.html.
[24] Solomon Hykes and others. What is Docker? https://www.docker.com/whatisdocker/.

[25] Advanced multi layered unification filesystem. http://aufs.sourceforge.net, 2014.
[26] VirtualBox http://www.virtualbox.org
[27] OvS, Open vSwitch, http://openvswitch.org/