Nadeem F, Qaiser R. An early evaluation and comparison of three private cloud computing software platforms. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(3): 639–654 May 2015. DOI 10.1007/s11390-015-1550-1

An Early Evaluation and Comparison of Three Private Cloud Computing Software Platforms

Farrukh Nadeem[1] and Rizwan Qaiser[2]

[1] *Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University Jeddah 21589, Saudi Arabia*

[2] *Department of Computer Science, National University of Computer & Emerging Sciences, Lahore 54500, Pakistan*

E-mail: fabdullatif@kau.edu.sa; rizwanqaiser.nu@gmail.com

Received June 18, 2014; revised January 19, 2015.

Abstract Cloud computing, after its success as a commercial infrastructure, is now emerging as a private infrastructure. The software platforms available to build private cloud computing infrastructure vary in their performance for management of cloud resources as well as in utilization of local physical resources. Organizations and individuals looking forward to reaping the benefits of private cloud computing need to understand which software platform would provide the efficient services and optimum utilization of cloud resources for their target applications. In this paper, we present our initial study on performance evaluation and comparison of three cloud computing software platforms from the perspective of common cloud users who intend to build their private clouds. We compare the performance of the selected software platforms from several respects describing their suitability for applications from different domains. Our results highlight the critical parameters for performance evaluation of a software platform and the best software platform for different application domains.

Keywords cloud computing, cloud computing software platform, performance evaluation and comparison

1 Introduction

Virtualization technology has changed the modern computing world. Rather than installing applications directly on physical machines, applications and system softwares, such as operating system, are installed on virtual machine (VM) images, which are then executed on physical servers through a hypervisor.

The cloud technology enables an easy and cost effective access to a large pool of well maintained virtualized resources (such as software, hardware, and development platforms provided as services) that are requested on demand. These resources are flexible to be dynamically reconfigured to adjust variable load and thus allow for optimum resource utilization[1].

So far, the cloud technology and associated resources have been mainly provided on commercial basis by organizations like Amazon①, Rackspace②, Google③, Microsoft④, referred to as "public clouds". However, there are a number of concerns about using public clouds. Some of these are as follows. Credit card processing applications have security concerns that may be challenging to solve, and many other business applications may require higher levels of performance, quality-of-service, and reliability that are not guaranteed by a public cloud service provider[2]. Government and commercial organizations dealing with proprietary data, such as banks, have a clear barrier towards public clouds due to critical security policies and overall system flexibility. Besides, some other organizations, like academic research groups, may have budget problems in repeatedly accessing the public clouds. All these concerns drive towards implementing the cloud environment from dedicated resources already available in

Regular Paper

①Amazon Inc. Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/, Jan. 2015.

②Rackspace Ltd. Rackspace open cloud. http://www.rackspace.com/, Jan. 2015.

③Google Inc. Google cloud platform. https://cloud.google.com/, Jan. 2015.

④Microsoft Inc. Windows azure. http://www.windowsazure.com/en-us/, Jan. 2015.

the organization, referred to as private cloud. Such an implementation would allow to exploit cloud computing platform while keeping all data secure behind a firewall. Several organizations, like Xen, Cisco, and Proxmox offer software support (referred to as cloud computing software platform (CCSP)) to implement private cloud computing environment.

The performance of a private cloud mainly depends upon the CCSP used to implement the cloud environment. A CCSP comprises of two major components: a cloud toolkit and a hypervisor. The cloud toolkit is responsible for providing necessary cloud functionality and overall management of the cloud resources. A hypervisor enables the virtualization technology to share and manage the local physical resources. It also underpins virtualization management, which includes real-time resource allocation, policy-based resource sharing, live migration, performance tuning, etc. For a given operating system, the performance of a CCSP depends upon the performance of the cloud toolkit as well as the hypervisor. The performance of the cloud toolkit depends upon the effectiveness and efficiency of underlying algorithms used to manage the cloud resources. The performance of a hypervisor is mainly driven by its architecture for hosting the guest operating systems. Consequently, the CCSPs with different cloud toolkits and hypervisors vary in their performance. Achieving maximum performance from private cloud resources requires evaluation and comparison of the CCSPs for target applications. For a common private cloud user, it is a difficult and cumbersome job to determine the CCSP that will deliver the optimum performance for his/her target applications.

To facilitate the common private cloud users (we do not target at the scientists/researchers), in this paper, we present the experimental results of our initial study on performance evaluation and comparison of three CCSPs: Ubuntu Enterprise Cloud (UEC)[5], Xen Cloud Platform (XCP)[6], and Proxmox virtual environment (PVE)[7].

We design our experiments to target at the evaluation of selected CCSPs' performance for a set of core applications from different domains, such as web serving, cryptography, database query processing, data compression, and so on. In current study, we evaluate the selected CCSPs for six different performance parameters, each targeting at a specific aspect of CCSP performance. These performance parameters are measured in our experiments using well-known benchmarks for each parameter. The results of our experiments highlight the critical parameters for performance evaluation of a CCSP. We also present a comparative view of the three CCSPs and describe which CCSP delivers the optimum performance for different domains of applications. Although our study is focused on facilitating common cloud users in selecting a CCSP that better suits his/her needs, our results provide the initial platform for the researchers for further in-depth studies of the selected and other CCSPs.

The rest of the paper is organized as follows. Section 2 sets the stage by describing basic models of cloud computing. In Section 3, we briefly describe the three CCSPs we evaluate in current study. The details of our experiments including performance evaluation model, performance parameters, benchmarks, and our experimental setup are described in Section 4. Section 5 describes the related work. We summarize and conclude our findings in Section 6. Finally, in Section 7, we describe our plans for the future work.

2 Cloud Computing Deployment Models

There are three basic models of cloud, namely public cloud, private cloud, and hybrid cloud. Each model has its own trade-offs and suitability for different needs. For example, applications needed infrequently best suit for deployment in a public cloud for its ready availability, cost effectiveness, etc. In contrast, applications needed on regular basis, or those having specific requirements on quality of service (QoS), location of data, security, etc. may best suit for deployment in a private or hybrid cloud[8]. In Subsections 2.1, 2.2, and 2.3, we briefly describe these cloud models.

2.1 Public Cloud Model

In a public cloud, the resources (compute resources, storage, infrastructure, etc.) are provided by the third parties over Internet as a service, through a pay-per-use model. The burden of resource configuration and maintenance completely rests on the resource providers, and

[5]Canonical Group. Ubuntu enterprise cloud. http://cloud.ubuntu.com, Jan. 2015.

[6]XEN Corp. Xen cloud platform. http://www.xen.org/products/cloudxen.html, Jan. 2015.

[7]Proxmox Server Solutions. Proxmox virtual environment. http://pve.proxmox.com, Jan. 2015.

[8]http://www.techrepublic.com/resource-library/whitepapers/introduction-to-cloud-computing-architecture/, Mar. 2015

the clients get a ready-made and easy-to-use infrastructure in place. Thus, this model greatly simplifies the use of cloud technology and also helps avoid the need of purchasing additional resources and the human expertise to manage these resources. At the same time, this model also has some drawbacks. For instance, the clients have little or no control over the underlying infrastructure — the applications from several clients may be executed simultaneously on shared resources, which may make it very difficult to meet certain QoS requirements. Likewise, to exploit public clouds, the client data has to be transferred to and stored at the servers of the service provider, which raises serious questions about the security of the data.

2.2 Private Cloud Model

A private cloud (also called an internal or enterprise cloud) offers the same services and benefits as the public clouds. However, it is built exclusively for an enterprise (usually from the enterprise's internal resources) providing complete control over the cloud infrastructure. Thus, the private cloud model removes several objections to the public cloud model including worries about data security, restrictions of network bandwidth, possible issues associated with legal requirements, QoS, etc. At the same time, the enterprise owning the private cloud bears the whole cost and running expenses of the private cloud. The private cloud model is more suitable for highly-regulated organizations and enterprises with requirements on proprietary data control. In addition, a private cloud offers an enterprise the ability to quickly develop and test cloud-aware applications behind its firewall. This includes the development of privacy-sensitive applications such as medical record database, credit card processing, classified data handling, and so on.[3]

2.3 Hybrid Cloud Model

The above two models of cloud computing have a fundamental trade-off between simplicity and privacy. For the enterprises requiring both – the simplicity of the public cloud as well as the data privacy and security offered by the private cloud – the hybrid cloud model might be the ideal choice. The hybrid cloud is a combination of interoperating public and private clouds. In this model, an organization (having a small private cloud, but requiring more resource to meet all of its needs) typically outsources the non-sensitive part of data/applications/services for processing to a public cloud, while exploiting its private cloud for processing the sensitive parts of data and applications/services. The aim of such a hybrid infrastructure is to leverage the benefits of on-demand scalability of public clouds while maintaining the required security and QoS through private cloud.

3 Cloud Computing Software Platforms

In this section, we describe the three CCSPs (UEC, XCP, PVE) that we consider in our current study. The reasons for selecting these three CCSPs are multifold. First, these CCSPs are free and well supported, and thus attract a large community. Second, these CCSPs have different hypervisors integrated with them: KVM[9] integrated with UEC, Xen integrated with XCP, and OpenVz integrated with PVE. Other well-known CCSPs like OpenStack, CloudStack, OpenNebula, either have one of these hypervisors or support them. Thus, we confine our current study to these three hypervisors only. It should be noted that the three hypervisors are default hypervisors that come with the three CCSPs (further details are presented in following subsections). Although the selected CCSPs also support other hypervisors, they show the best performance with their default hypervisors because their codes are tuned for their default hypervisors. Therefore, we evaluate their performance with the default hypervisors only. Third, we plan to have a first-hand small study to investigate if there are significant performance differences among these three. In case of positive answer, further detailed study may be planned.

3.1 Ubuntu Enterprise Cloud

Ubuntu Enterprise Cloud consists of Ubuntu Server Edition, with Eucalyptus[4] (that uses the KVM hypervisor in our study) and libvirt[10]. Eucalyptus is an open core (a version also available as open source) software platform for implementing private and hybrid clouds. It enables enterprises to turn their existing infrastructures including computers, networks, and storage into their own private cloud that is controlled and customized locally. It provides Amazon EC2[11] like infrastructure

[9]Redhat. Kernel based virtual machine (KVM). http://www.linux-kvm.org/page/Main_Page, Jan. 2015.

[10]Red Hat. Libvirt, the virtualization API. http://libvirt.org/, Jan. 2015.

[11]Amazon Inc. Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/, 2008, Jan. 2015.

capabilities by implementing Amazon EC2 Application Programming Interface (API).

A typical Eucalyptus architecture is shown in Fig.1. Eucalyptus consists of five major components that interact in a hierarchical manner. A *cloud controller* is the user visible entry point as well as the global decision making component of Eucalyptus. It is responsible for processing administrative policies and user initiated requests, processing service level agreements, maintaining persistent system level and user requested metadata, and making high-level VM instance scheduling decisions and forwarding them to the cluster controllers. A cluster controller typically executes on a cluster head node (also referred to as server) that has access to both private and public networks. It takes requests from the cloud controller and gets them done on the compute nodes (the physical machines, which are designed to execute user requests through VM instances). It is also responsible for gathering state information from its collection of compute nodes, scheduling incoming execution requests to individual compute nodes, and managing the configuration of public and private networks. The node controller executes on each compute node. It queries the compute nodes to discover their physical resources, like the number of CPUs and cores, the size of memory, and the available disk space.

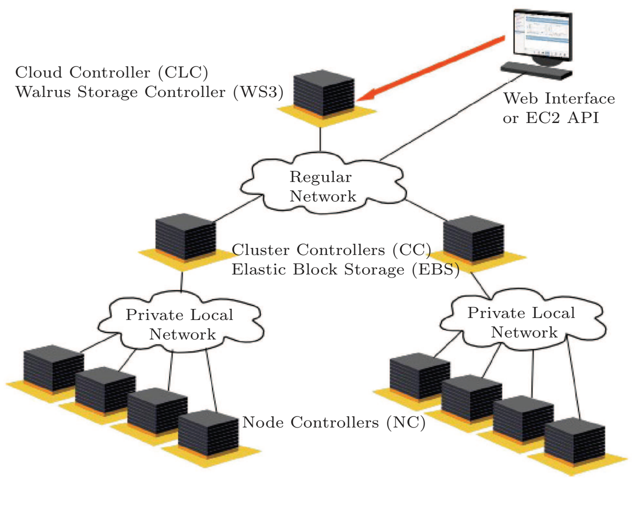

Fig.1. Typical Eucalyptus architecture[3].

Its role includes VM start-up, query, shutdown and clean-up in response to the requests coming from its immediate cluster controller. The *elastic block storage controller* runs on the same machine(s) as the cluster controller and allows to create persistent block devices that can be mounted on running machines in order to gain access to virtual hard drive[4].

3.2 Xen Cloud Platform

The Xen Cloud Platform is an open source cloud computing software platform. It typically includes Xen hypervisor, Xen management API (called XAPI), and a variety of virtual infrastructure cloud services like security, storage, and network virtualization. XCP cloud services can be leveraged to enable isolation and multi-tenancy capabilities in cloud environments. Fig.2 depicts a typical architecture of XCP. The Xen hypervisor is the first program that runs when XCP boots, i.e., right after the local bootloader. It runs directly on the host hardware and virtualizes the host's physical resources like CPU, interrupts, and memory. The VM instances run on top of Xen. A running instance of VM in Xen is called a domain. A special domain, called control domain (or domain 0), boots next. Domain 0 is basically a modified Linux kernel that runs as an XCP VM with additional privileges of controlling the host hardware devices. It includes XAPI toolstack, which provides management functionalities for creation, configuration, and destruction of guest VMs. The XAPI toolstack provides an interface to the outer world that can be accessed through a command line console, a graphical user interface, and a cloud orchestration stack such as OpenStack or CloudStack.

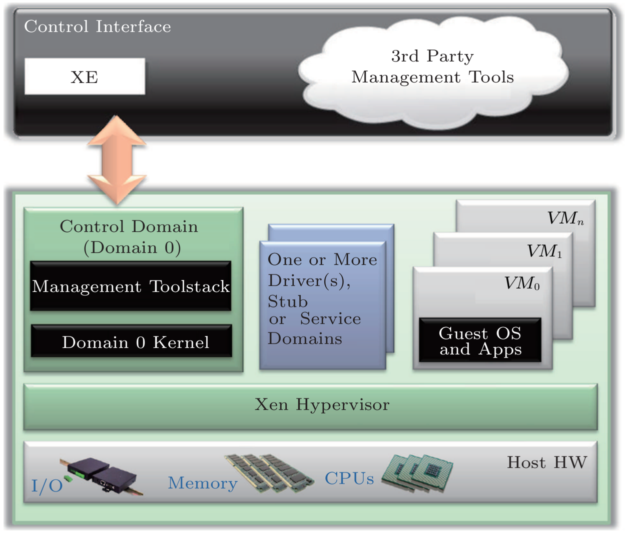

Fig.2. Xen Cloud Platform architecture (adapted from "Xen Overview"[12]).

[12]XEN. XEN overview. http://wiki.xen.org/wiki/Xen_Overview, Jan. 2015.

Xen allows the simultaneous execution of multiple paravirtualized guest VMs (probably running different operating systems) on the same host, each of which provides entirely isolated computation, storage, and networking to the operating system running inside it. Xen also allows the main device drivers for a system to run inside a virtual machine.

This enables an easy restart of the drivers (in case of their failures/updates), through rebooting the VM containing the driver without affecting the rest of the system. XCP includes a command-line interface "XE" to forward user requests to XCP hosts.

3.3 Proxmox Virtual Environment

Proxmox Virtual Environment is an open source server virtualization platform. It is based on Debian 6 Squeeze at 64 bits, which allows to create a virtualization environment of type "bare" metal based on full virtualization by KVM as well as container virtualization by OpenVZ (in our case). OpenVZ is a virtualization technology based on the Linux kernel. It allows the creation of multiple isolated containers (or virtual environments (VEs)) on a single physical server enabling better server utilization. Different containers can run different Linux distributions, while they all operate under the same kernel. Each VE executes exactly like a stand-alone server — it has its own set of processes starting from *init*, users (including root), IP addresses, file system, routing tables, network interfaces, etc. and can be restarted independently.

The architecture of OpenVZ is shown in Fig.3[13]. PVE cluster connects multiple physical nodes together to form a multi-master cluster environment (i.e., no single point of failure). The cluster mode enables central management, load balancing, live migration of VEs, etc.

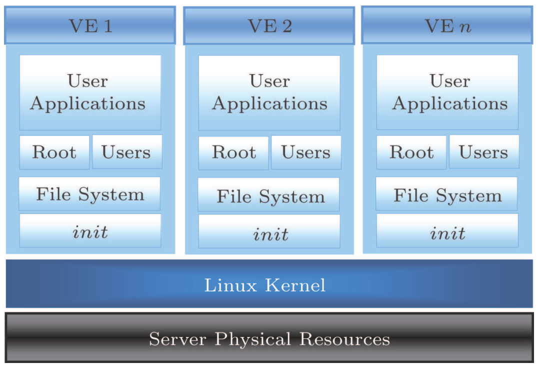

Fig.3. OpenVZ architecture[13].

4 Performance Evaluation of Cloud Computing Software Platforms

In this section, we describe our performance evaluation model, the performance parameters, and the respective benchmarks considered in our evaluation and the setup for our experiments. In addition, we present and discuss findings from our experiments.

4.1 Performance Evaluation Model

Our performance evaluation model is based on traditional system benchmarking for individual components as well as the whole system. It comprises five major steps as shown in Fig.4. We start by selecting "infrastructure as a service" model of the cloud and build the cloud environment from each CCSP for the same set of hardware resources.

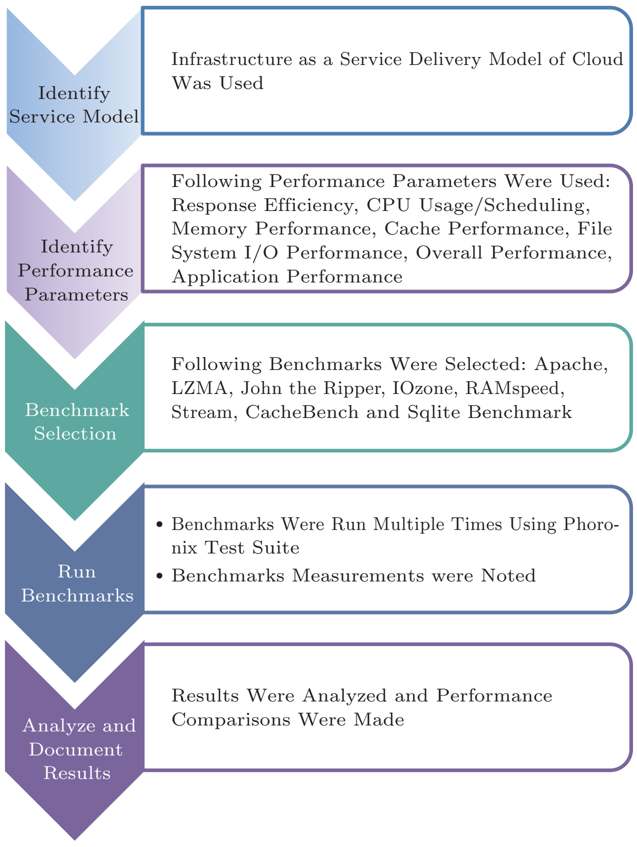
Identify Service Model
Infrastructure as a Service Delivery Model of Cloud Was Used
Identify Performance Parameters
Following Performance Parameters Were Used: Response Efficiency, CPU Usage/Scheduling, Memory Performance, Cache Performance, File System I/O Performance, Overall Performance, Application Performance
Benchmark Selection
Following Benchmarks Were Selected: Apache, LZMA, John the Ripper, IOzone, RAMspeed, Stream, CacheBench and Sqlite Benchmark
Run Benchmarks
• Benchmarks Were Run Multiple Times Using Phoronix Test Suite
• Benchmarks Measurements were Noted
Analyze and Document Results
Results Were Analyzed and Performance Comparisons Were Made

Fig.4. CCSP performance evaluation model.

In the second step, we select six parameters to evaluate the performance of our selected CCSPs. The motivation for selecting these parameters comes

[13]http://www.sparksupport.com/blog/server-virtualization, Apr. 2015.

from our target of the current study to answer the following major questions from the perspective of users/owners/administrators of private clouds.

• How quickly will a CCSP respond to a user request?

• How effectively and efficiently will the CCSP exploit the critical hardware resources like CPU and memory?

• Which CCSP will deliver the optimum performance for target application?

It has already been shown that the performance evaluation of various individual components also provides a first order estimate of that system's performance[5]. Our selected parameters are: 1) response efficiency, 2) throughput, 3) memory performance, 4) cache performance, 5) file system I/O performance, as well as 6) an application performance in a CCSP. These parameters are briefly described in Subsection 4.2. Since the enterprises usually have dedicated high speed LAN without any restrictions of bandwidth, we do not include network evaluation parameters in current study.

In the third step, we select the benchmarks to measure the selected parameters through our experiments. The considerations for choosing the benchmarks are as follows. First, the benchmark should be well known and widely accepted to evaluate the target parameters so that our results become trustworthy for a wide range of community. Second, the benchmarks from different providers should be selected so that our results are not restricted to one class of algorithms provided by one benchmark provider. Third, the benchmarks should show reproducible results under a given set of conditions. The benchmarks for their respective parameters are also described along with the parameters description in Subsection 4.2. In the fourth step, we execute these benchmarks on our experimental platform (see Subsection 4.3). In the last step, we analyze the results of our experiments.

4.2 Performance Parameters and Benchmarks

This subsection briefly describes the parameters and the corresponding benchmarks considered in current study.

1) *Response Efficiency*. Response efficiency of a cloud platform is a measure of its readiness to respond to a request and is of critical importance for several types of applications, for example, real time applications, web applications. It is commonly measured in terms of response time, which is the elapsed time between the end of the request to a cloud platform and the beginning of its response. In our experiments, to evaluate response time, we use Apache benchmark[14] for responding to a request of a static webpage.

2) *CPU Throughput*. CPU throughput is a measure of the system's computational work done per unit time. It is also an evaluation of CPU scheduling policies of CCSPs. We measure CPU throughput of the CCSPs to evaluate how effectively their CPU management subsystems utilize the most important hardware resource, i.e., the CPU, which is very critical, especially for CPU intensive applications. The CPU throughput is evaluated through *John the Ripper* benchmark[15]. *John the Ripper* is a popular password testing/breaking program that employs different ciphers, like DES and MD5. In our experiments, *John the Ripper* tries different username/password combinations to crack in.

3) *Memory Performance*. We evaluate the memory performance of a CCSP to evaluate the effectiveness of their memory management subsystems in utilizing the main memory. The main memory management is of critical importance, especially for the applications that load large datasets into the memory during their execution, like data intensive applications. To evaluate memory performance of a CCSP, we consider the maximum memory bandwidth achieved during different memory operations. Memory bandwidth is evaluated using RAMspeed[16] benchmark. RAMspeed is a memory intensive benchmark that measures the memory bandwidth for read, write and data manipulation memory operations. To this end, RAMspeed performs numerical operations of *copy, scale, add*, and *triad* using different large blocks of data. The *copy* operation (e.g., $X = Y$) replicates data from one memory location to another. The *scale* operation modifies the data, before writing to a memory location, by multiplying with a certain value, (e.g., $X = n \times Y$). The *add* operation reads data from two different memory locations, adds them up and writes the result to a third memory location (e.g., $X = Y + Z$). The *triad* operation reads data from a memory location, scales it, adds data from another memory location, and writes the result to a third

[14]The Apache Software Foundation. Apache http server benchmarking tool. http://httpd.apache.org/docs/2.2/programs/ab.html, Jan. 2015.

[15]Openwall. John the Ripper password cracker. http://www.openwall.com/john/, Jan. 2015.

[16]RAMspeed: A cache and memory benchmarking tool. http://alasir.com/software/ramspeed/, Jan. 2015

memory location (e.g., $X = n \times Y + Z$). In another test, RAMspeed performs *copy, scale, add* and *triad* operation in a single run and measures overall bandwidth in these operations. In our experimental results, this bandwidth is referred to as "average".

4) *Cache Performance*. Many scientific applications have significant resource requirements in terms of memory footprint[6]. High speedups of these applications are often achieved by exploiting the cache. This is especially true given the widening gap between processor speed and main memory. The cache performance provides a good basis for evaluating a CCSP for those applications that have already been substantially tuned for cache reuse[7].

In order to evaluate the performance of cache memory, we use "CacheBench"[17]. CacheBench evaluates the performance of possibly multiple levels of cache present on a system. The goal of this benchmark is to establish peak computation rate given optimal cache reuse. CacheBench incorporates benchmarks for different cache operations like *cache read, cache write*, and *cache read/write/modify*. Each of these benchmarks performs repeated access to data items of varying vector lengths, gathers timings over iterations, and computes a cache bandwidth (in MB per second) by dividing the total amount of data by the total time.

5) *File System I/O Performance*. Many scientific applications, like MeteoAG[8], read from and write a lot of data to the disk. The performance of a file system has a key role in the overall performance of such applications. We evaluate the file system I/O performance of the selected CCSPs using "IOzone" benchmark. IOzone is a file system benchmark tool that measures a variety of file system operations, like *disk read, disk write* or something else.

6) *Overall Performance of an Application in a CCSP*. Besides evaluating individual performance aspects of a CCSP, it is also vital to evaluate its overall performance for target applications. We evaluate the selected CCSPs for two types of applications: first, a general compression application; second, a database application. The performance of general compression application is evaluated in terms of its execution efficiency (measured in terms of execution time). For this purpose, we consider "LZMA" compression benchmark. The LZMA (Lempel−Ziv−Markov Chain Algorithm) is a compression algorithm used to perform lossless data compression.

To evaluate the performance of a database application in a CCSP, we consider its number of transactions completed per second (TPS). We define TPS of an application as its number of atomic operations performed in a second. In our experiments, TPS is measured using "Sqlite" benchmark[18]. Sqlite benchmark performs 12 500 insert operations into an indexed database. The TPS is calculated by the total time of insert operations divided by 12 500.

4.3 Experimental Setup

We now describe the experimental setup we use for the performance evaluation of the CCSPs presented in Section 3.

Environment. We conduct our experiments on a private 10-node homogeneous cluster, where each system is equipped with Intel 2.93 GHz processor, 5 GB memory, and 80 GB disk space. Ubuntu 10.04 LTS (64-bit) image is used in all three cloud platforms. We use a modified *m*1*.large* instance type (an instance type of Amazon EC2, which is essentially a VM) that uses 20 GB disk space and 512 MB RAM. The performance of Ubuntu enterprise cloud, Xen cloud Platform, and Proxmox-VE is evaluated with KVM, Xen, and OpenVz hypervisors respectively.

Performance Benchmarking Tool. To ensure the consistency in our experiments, we conduct our experiments through Phoronix test suite[19]. The Phoronix test suite is a comprehensive testing and benchmarking platform that is designed to effectively carry out both qualitative and quantitative benchmarks in a clean, reproducible, and easy-to-use manner.

4.4 Experimental Results

In this subsection, we describe the empirical evaluation of the selected CCSPs. We repeat our experiments several times and the average values of the results (along with the minimum and maximum values in some cases, to show the variability in the results) are shown in this paper. It should be noted that we do not aim at the in-depth evaluation of algorithms implemented in each subsystem of a CCSP — that is beyond the scope of this study.

[17] CacheBench benchmark. http://icl.cs.utk.edu/projects/llcbench/cachebench.html, Jan. 2015.

[18] Sqlite. http://www.sqlite.org/index.html, Jan. 2015

[19] Phoronix Media. Phoronix test suite. www.phoronix-test-suite.com/, Jan. 2015.

Response Efficiency. The performance metric measured in our experiments for the evaluation of response efficiency, is the response time of Apache web server to handle a request of a static page (the lower, the better). Fig.5 shows the response time of Apache web server to a request of a static page, for the three CCSPs. As Apache web server's time to react to a given request is very short (in microseconds (ms)), for demonstrative and comparison purposes, we also present it in terms of the number of static page requests handled per second by Apache web server — the higher the number of page requests handled, the more efficient the system. Fig.6 shows the number of requests handled per second by the Apache web server in the three CCSPs. PVE shows the smallest response time of the three CCSPs (average: 97 ms, min.: 95 ms, max.: 98 ms) and thus appears to be the most efficient of the three CCSPs, whereas XCP shows the highest response time (average: 200 ms, min.: 197 ms, max.: 202 ms). Correspondingly, PVE handles approximately twice the number of requests (10 353 on average) per second compared with XCP (508 on average). A small variability (shown as standard deviation bars in Fig.5 and Fig.6) is observed in the performance of PVE and XCP (0.008%), whereas that of UEC is the highest of the three (18%).

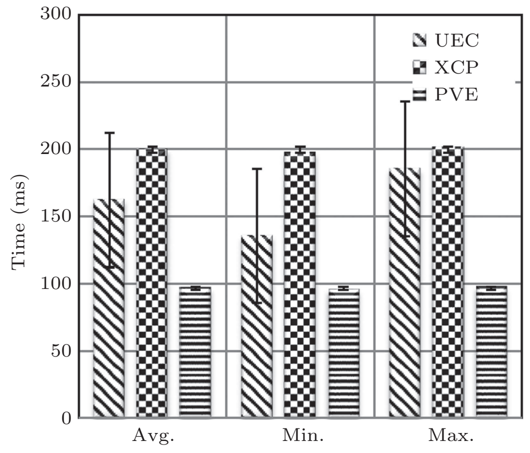

Fig.5. Response efficiency: Apache benchmark results (response time).

CPU Throughput. The performance metric for our CPU throughput experiments is the number of username/password combinations processed per second by *John the Ripper* (the higher, the better). We conduct our experiments for three different ciphers in *John the ripper* namely: DES, MD5 and blowfish. Figs.7(a)∼7(c) show the number of username/password combinations processed per second for each of the three CCSPs. In all of the experiments for the three ciphers, PVE processes the highest number of username/password combinations (thus the highest CPU throughput) and UEC processes the lowest (thus the lowest CPU throughput). For DES cipher, PVE produces 0.7% and 0.4% higher throughput than UEC and XCP respectively. For MD5 cipher, PVE produces 0.8% and 0.6% higher throughput than UEC and XCP respectively. For blowfish cipher, PVE produces 0.7% and 0.3% higher throughput than UEC and XCP respectively. On average, PVE yields 0.7% and 0.4% higher throughput than UEC and XCP respectively. We observe the consistent performance of the three CCSPs in all of our experiments for CPU throughput — very small variability of at most 0.2%.

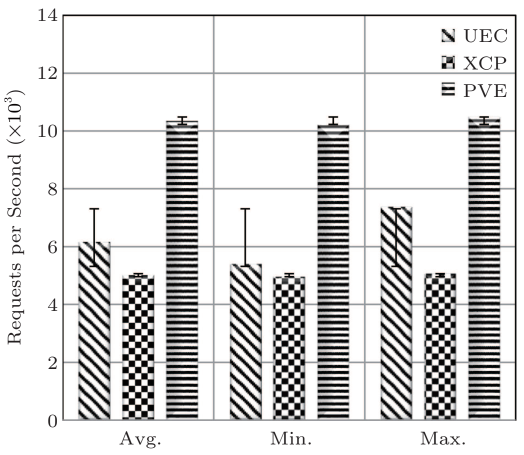

Fig.6. Response efficiency: Apache benchmark results (number of requests per second).

Memory Performance. The performance metric for our memory performance experiments, which are based on RAMspeed benchmark, is the memory bandwidth achieved in different memory operations (the higher, the better). In our RAMspeed benchmarking experiments, we consider memory operations for floating-point and integer data types, because the majority of scientific applications use these data types. Figs.8(a) and 8(b) depict the RAMspeed benchmark results (in terms of memory bandwidth) for different memory operations in the three selected CCSPs, for integer and floating-point data type respectively. For integer data type, UEC demonstrates the lowest bandwidth in all the memory operations. XCP demonstrates a little higher memory bandwidth than PVE for *copy* and *scale* memory operations (higher by 31 MB/s and 29 MB/s respectively), whereas PVE supersedes XCP for *triad*, *add*, and *average* memory operations

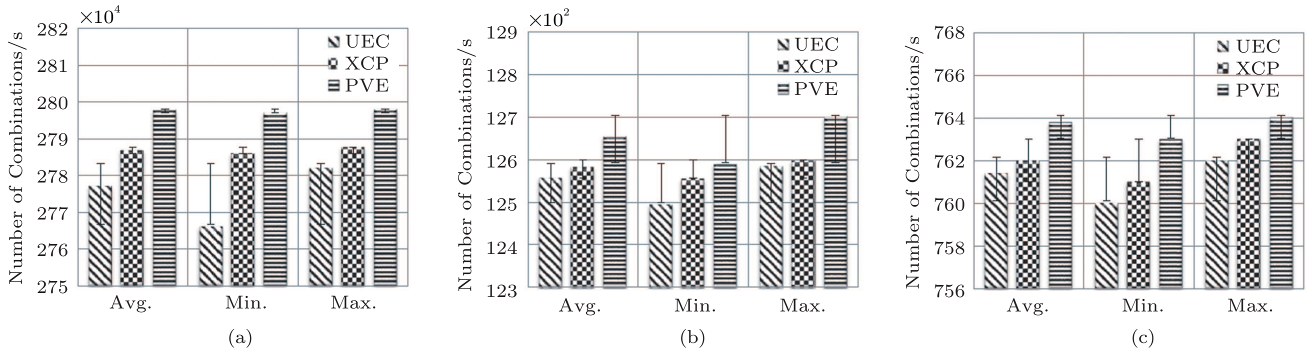

Fig.7. CPU throughput: *John the Ripper* benchmark results (number of username/password combinations processed per second) for the three CCSPs. (a) DES cipher. (b) MD5 cipher. (c) Blowfish cipher.

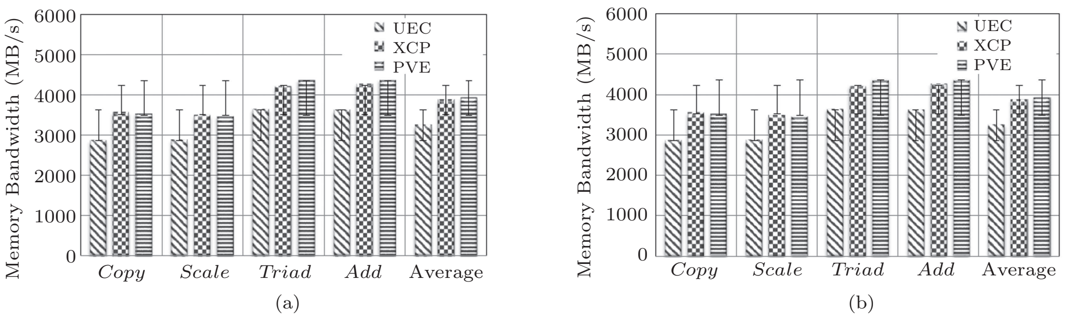

Fig.8. Memory performance: RAMspeed benchmark results (memory bandwidth achieved during different memory operations) for integer and floating-point data types in the three CCSPs. (a) Memory operations (integer data type). (b) Memory operations (floating-point data type).

(higher by 158 MB/s, 104 MB/s and 50 MB/s respectively). We observe similar results for floating-point memory operations. The three CCSPs show consistent performance in memory operations (a small variability of 0.1% is observed).

Cache Performance. In our experiments, we benchmark the cache performance of the three CCSPs for:

• *cache read*: provides read bandwidth for varying vector lengths in a compiler optimized loop;

• *cache write*: provides write bandwidth for varying vector lengths in a compiler optimized loop;

• *cache read/modify/write*: generates read/modify/write bandwidth for varying vector lengths in compiler-optimized loop operations.

The performance metric for cache performance experiments (through CacheBench benchmark) is the cache bandwidth achieved in different memory operations (the higher, the better). Figs.9(a)∼9(c) show the cache bandwidth achieved by the three CCSPs in *read, write*, and *read/modify/write* cache operations respectively. In case of *read* and *read/modify/write* cache operations, PVE outperforms the other two CCSPs. However, in case of *write* cache operation, XCP supersedes the other two. In case of *read* and *write* cache operations, the average cache bandwidth and the maximum cache bandwidth in UEC and XCP are very close to each other. PVE exhibits the highest (of the three CCSPs) variability of the cache bandwidth. Whereas, XCP exhibits the most consistent bandwidth.

File System I/O Performance. We conduct our file system I/O performance evaluation experiments with IOzone benchmark for:

• *disk read*: provides the bandwidth of reading a file from the disk for varying record sizes;

• *disk write*: provides the bandwidth of writing a new file to the disk for varying record sizes.

The performance metric for file system I/O performance experiments is the bandwidth achieved in file system I/O operations (the higher, the better). We conduct our experiments for a file size of 512 MB, with three different record sizes 4 KB, 64 KB and 1 MB.

Figs.10(a)∼10(c) depict the bandwidth achieved for *disk read* operation for the three record sizes respectively. The highest read bandwidth is observed during our experiments with UEC, whereas the experiments with XCP yield the lowest read bandwidth. On average, our experiments with UEC yield 65% higher *disk read* bandwidth than PVE. We observe a serious perfor-

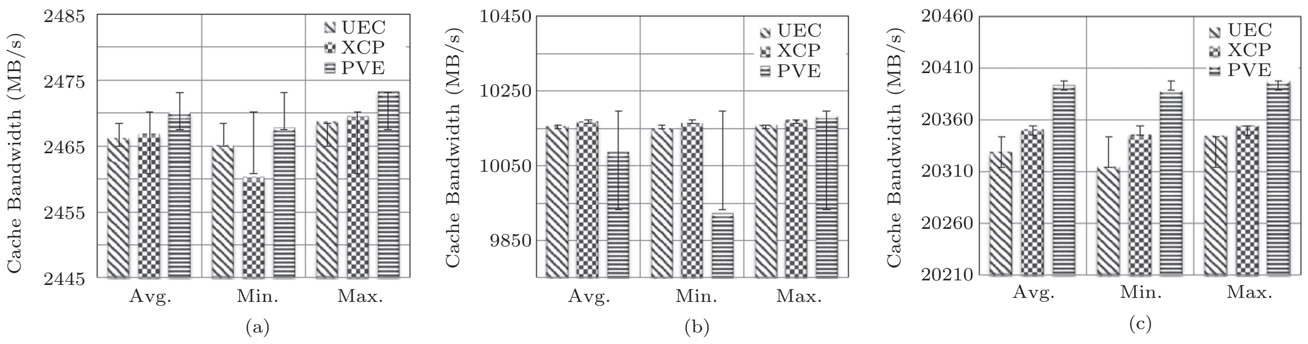

Fig.9. Cache performance: CacheBench benchmark results (cache bandwidth achieved during different memory operations). (a) *Cache read* operation. (b) *Cache write* operation. (c) *Cache read/modify/write* operations.

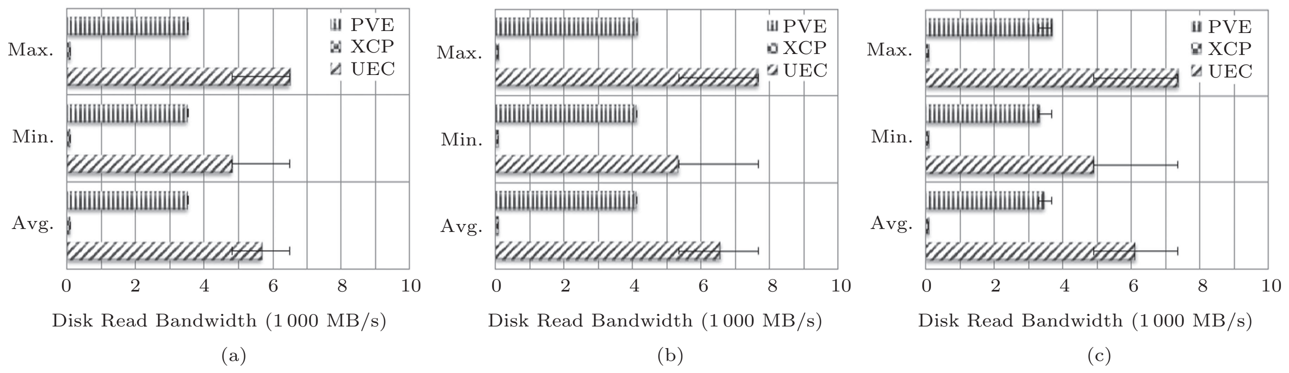

Fig.10. File system I/O performance: IOzone read benchmark results (read bandwidth achieved for three block sizes). (a) File size = 512 MB, record size = 4 KB. (b) File size = 512 MB, record size = 64 KB. (c) File size = 512 MB, record size = 1 MB.

mance bottleneck in case of XCP that restricts its performance. Further in-depth experiments are required to investigate the reasons for this bottleneck and are beyond the scope of this study. On the other hand, UEC shows 14% variability (standard deviation) over the three block sizes, whereas the other two CCSPs exhibit a small variability (less than 3%).

Figs.11(a)~11(c) depict the bandwidth achieved in *disk write* operation for the three record sizes respectively. This time PVE outperforms the other two CCSPs and shows the highest write bandwidth, while UEC considerably degrades in write performance and shows the lowest write bandwidth. On average, we find that the write operation in PVE is 8 times (approx.) faster than that in XCP. It is noteworthy that the three CCSPs exhibit consistent write performance (i.e., very small variability, less than 1%) for each of the block sizes.

We observe PVE showing a consistent change in *disk read* and *disk write* operations when the block size is varied. The bandwidth (both read and write) is increased when the block size is increased from 4 KB to 64 KB and decreased when the block size is further increased from 64 KB to 1 MB. However, the performance of the other two CCSPs does not change with block size. Furthermore, as expected, a clear difference between read and write bandwidths of each CCSP is observed — the *read* operation is much faster than the *write* operation.

Application Performance in a CCSP. The performance metric for our experiments for the evaluation of a general compression application in each CCSP is the amount of time (the smaller, the better) that is consumed in compressing a file using "LZMA" compression algorithm. Fig.12 depicts the execution time of LZMA file compression. We do not observe large differences in the performance of the three CCSPs. Nevertheless, XCP takes the shortest time and PVE takes the longest time.

Fig.13 shows the TPS (the higher, the better) in our Sqlite benchmark experiments. PVE and UEC process the highest and the lowest number of transactions per second respectively. On average, Sqlite's TPS for PVE is 27 times and 47% more than that of UEC and XCP respectively. It is noteworthy that the ranking of the three CCSPs through Sqlite benchmark is the same as

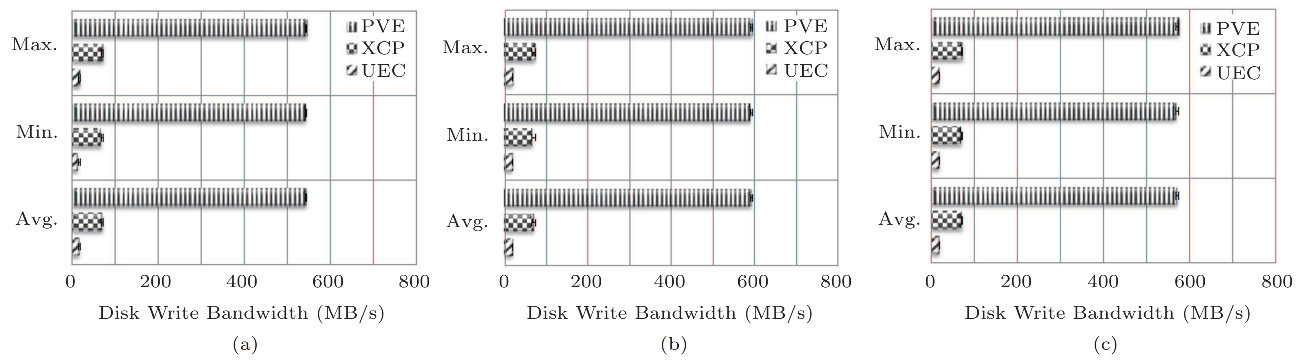

Fig.11. File system I/O performance: IOzone write benchmark results (write bandwidth achieved for three block sizes). (a) File size = 512 MB, record size = 4 KB. (b) File size = 512 MB, record size = 64 KB. (c) File size = 512 MB, record size = 1 MB.

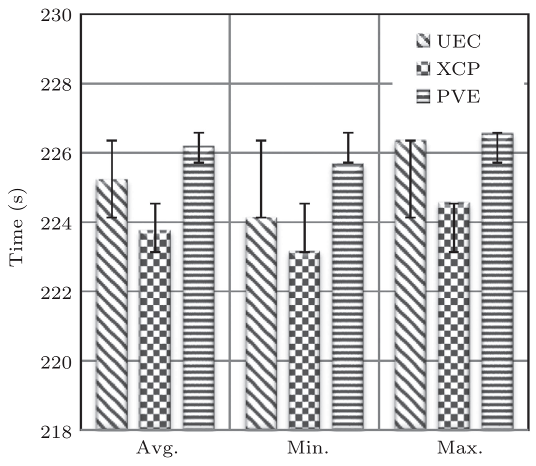

Fig.12. LZMA benchmark results.

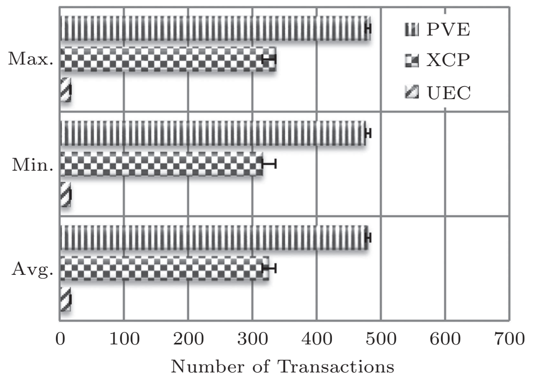

Fig.13. Transactions per second.

that through IOzone-write. It is mainly due to the large involvement of disk write operation within the Sqlite benchmark experiments. However, compared with IOzone experiments, the relative performance of XCP increases in Sqlite experiments. In further investigative experiments, we find that this improvement in the performance is due to the fact that Sqlite bypasses the local file system for its *disk read* and *disk write* operations, and thus eliminates the performance bottleneck due to the local file system. However, this improvement is not significant in case of UEC.

Our further experiments reveal that this difference in performance behavior is due to that architectures of KVM, XCP and PVE – KVM use full virtualization[5], XEN uses paravirtualization[20], and PVE uses container virtualization. Full virtualization creates a complete virtual system by providing the total abstraction of the underlying physical hardware. No modification is required in guest OS/application and it executes in the virtual system as it executes on underlying physical hardware. Thus the guest operating system is not aware of that it executes in a virtualized system. Paravirtualization creates a modified abstraction of underlying physical hardware. In paravirtualization technique, the guest OS is modified to improve its performance by replacing its non-virtualizable instructions with hypercalls that are sent directly to the hypervisor. Thus, the guest OS knows that it is executing on a virtual machine. The container-based (also called OS-level) virtualization does not have a separate hypervisor. Instead, the virtualization layer is a part of the host OS and provides all the functions of a hypervisor. The isolated guest virtual machines (called containers) run on the top of host kernel and thus only one operating system takes care of hardware calls. In turn, this approach requires that guest virtual machines must have the same OS as the host. The average performance variance in all experiments for XCP, UEC, and PVE is 3%, 2%, and 1%, respectively.

[20]XEN. XEN overview. http://wiki.xen.org/wiki/Xen_Overview, Jan. 2015.

5 Related Work

Several studies have analyzed the performance of hypervisors. An experimental study on the performance interference in parallel processing of CPU and network intensive workloads in Xen VM monitors has been performed in [6]. Experiments were conducted to measure the performance interference among VMs running network I/O workloads that are either CPU-bound or network-bound. Study in [9] compared two core hypervisors, Xen and KVM. The most prominent difference between the two hypervisors was found in scalability. In addition, KVM was observed to have considerable problems of guests crashing with four or more guests. In contrast, KVM showed better isolation than Xen. In nutshell, it was observed that KVM outperformed Xen in I/O-intensive tests, while Xen outperformed KVM in CPU-intensive tests.

In contrast to these studies, we focus on complete CCSPs, not the hypervisors only. A study[21] on the evaluation of performance of virtual machine monitors in VMware compared with their counterparts in Xen showed that VMware outperformed Xen in this respect. A recent work[10] compared the performance of three hypervisors XenServer, VMware ESXi, and KVM, and found that XenServer and ESXi Server both performed equally, whereas KVM fell behind these two.

Much research has been done on the performance evaluation of single cloud environments or specific aspects of the cloud. For instance, Palankar *et al.*[11] analyzed the performance of Amazon S3 with reference to scientific community. They also discussed its security features and concluded that S3 was not ready to satisfy the needs of scientific community. The study in [12] evaluated the performance of a single cloud only. It is focused on how applications perform under the virtualized environment of Amazon EC2. Similarly, study in [13] targets at the evaluation of storage and communication performance of Illinois cloud computing testbed. Iosup *et al.*[14] analyzed the performance of EC2 cloud computing services from the perspective of finding if cloud computing offerings are sufficient for scientific computing demands. From their experiments, they found that the performance and reliability of current cloud setup was low. Garfinkel[15] analyzed Amazon EC2 for its APIs, availability, ease of use, management and security facilities. He also described an end-to-end performance analysis of S3 throughput and latency as observed from EC2 and other locations on the Internet. Study in [16] compares the performance of several storage and file systems for their impacts on the performance and cost of executing workflows on Amazon EC2. The work by Kobayashi *et al.*[17] evaluated the performance of Gfarm file system on a Eucalyptus-based private cloud. Their experiments with MapReduce applications as well as micro-benchmarks showed that Gfarm delivered better scalable I/O performance than the other file systems. Wang *et al.*[18] addressed the use of economy of scale benefits of cloud computing for small-to-medium scale scientific communities. The authors proposed a public cloud usage model for small-to-medium scale scientific communities to utilize elastic resources. The authors also presented DawningCloud to facilitate building lightweight management services for heterogeneous workloads. Their proposed model and system could save the total resource consumption by 54%.

A couple of studies investigated the performance of clouds for scientific applications and compared them with typical high performance computing (HPC) systems or Grids[19-24]. Walker[19] examined EC2's performance particularly for high performance scientific applications by using micro- and macro-benchmarks. He compared the performance of a cluster composed of EC2 high-CPU compute nodes with the performance of a cluster composed of equivalent processors available to the open scientific research community and found a significant performance gap in the examined clusters. In their study[20], Ostermann *et al.* analyzed the performance of the Amazon EC2 platform using micro-benchmarks and kernels. They concluded that the current cloud services need significant performance improvement to be useful to the scientific community. Jackson *et al.*[21] compared conventional HPC platforms with Amazon EC2 using real-world applications, which are representative of the workloads at a typical supercomputing center. They found that EC2 was twenty times slower than a modern HPC system, and six times slower than a typical mid-range Linux cluster. Gupta and Milojicic[25] compared the performance of HPC applications on three cloud platforms with HPC-optimized cluster. They concluded that the cloud was a viable platform for some applications, specifically, non-communication intensive applications such as embarrassingly parallel applications; it was not viable for communication-intensive applications for high processor count.

[21] VMWare. A performance comparison of hypervisors. http://www.vmware.com/pdf/hypervisor_performance.pdf, Mar. 2015.

A few studies focus on the performance evaluation of clouds with respect to workflow applications[26-29]. Juve *et al.*[26-27] examined Amazon EC2 with three real scientific workflow applications. Comparing EC2's performance with that of a typical HPC system, they concluded that cloud's performance comparable to HPC systems could be achieved given similar resources. Vecchiola *et al.*[28] compared the performance of Amazon EC2 using S3 storage against Grid'5000 by running fMRI brain imaging workflow application. In contrast to these studies, we target at the comparative evaluation of three private CCSPs where our focus is to rank them with respect to different parameters.

Li *et al.*[30] presented a comprehensive measurement study over four major cloud providers, namely, Amazon AWS, Microsoft Azure, Google AppEngine, and Rackspace CloudServers. They evaluated the computing, storage, and networking services offered by these cloud providers. In addition, they also considered specific metrics that directly reflect cloud services' impact on the performance of customer applications.

Some latest efforts compared the performance of public clouds with that of private clouds. In [31], Ward compared the performance of public cloud Amazon EC2 with a private cloud built using Ubuntu Enterprise Cloud (UEC). He ran the same benchmark suites on both cloud setups under same conditions to evaluate their performance. He found that UEC's performance equaled, and in some cases even surpassed EC2's performance. It was concluded that KVM requires further development in order to offer this performance to all virtualized applications, especially database applications, which do not perform well on UEC. Our experimental results for the performance evaluation of UEC are in accordance with this study. Tudoran *et al.*[32] evaluated performance of Azure[22] (public cloud) and a Nimbus[23] based private cloud for primary needs of scientific applications including computation power, storage, data transfer, and costs. Their results show that Nimbus incurs less variability and has increased support for data intensive applications, while Azure deploys faster and has a lower cost. Perhaps, the closest work to our present study is in [33], where authors compared the performance of Eucalyptus-based private cloud with VMware vCloud, Microsoft Cloud, and Citrix CloudStack. However, the comparison is based on static features. In contrast, our work targets at dynamic performance comparison of three CCSPs for private clouds.

The authors in [34] compared three open-source cloud computing software platforms, Eucalyptus, OpenNebula and OpenQRM for their security features. Based on their study, the authors suggested new security solutions for cloud platforms. The work in [35] presented a good framework for ranking cloud computing services. In their framework, the authors assigned different weights to different cloud evaluation parameters and computed the weighted sum of those parameters to rank the clouds. However, it is very difficult for a common user to evaluate cloud services individually and assign them numerical values to adopt the suggested framework.

In comparison with these studies, our study compares specifically private CCSPs under the same conditions with an objective of facilitating common cloud users to choose one CCSP that will deliver comparatively better performance for their target applications. Our experimental results highlight the differences in performance of the three clouds for different application domains. Our results are helpful in determining the most suitable CCSP for each application domain.

6 Conclusions

In this paper, we presented our study on performance evaluation and comparison of three open source cloud computing software platforms. We conducted extensive experiments to evaluate their performance using *m*1.*large* instance. Table 1 summarizes our findings from experiments with the three CCSPs.

Overall, from the selected performance parameters, PVE shows the best performance for response efficiency, CPU throughput, and application performance. UEC shows the best performance for file system I/O operations. No single CCSP shows the best performance in all memory and cache operations — in some operations, XCP appears to be the best while in some other ones, PVE dominates the other two; however, their performance differences are statistically insignificant.

Besides, during our experiments for disk operations, we observed that: *disk read* operation is much faster than *disk write* operation; XCP has some serious bottleneck for *disk read* operation; UEC performs the best for *disk read* operations, but for *disk write* operations,

[22]Microsoft Inc. Microsoft Azure. http://azure.microsoft.com/en-us/, Jan. 2015.

[23]University of Chicago. Nimbus infrastructure. http://www.nimbusproject.org/, Jan. 2015.

Table 1. Performance Comparison of the Three Cloud Computing Software Platforms

Serial	Performance Parameter	Performance Benchmark	Benchmark Operations	Performance Metrics	UEC	XCP	PVE
1	Response efficiency	Apache web server	—	Response time		👎	👍
			Performance consistency				👍
2	CPU throughput	*John the Ripper*	—	Number of username/password combinations	👎		👍
			Performance consistency			👍	
3	Memory	RAMspeed	*Copy*	Memory bandwidth	👎	👍	
			Scale		👎	👍	
			Triad		👎		👍
			Add		👎		👍
			Average		👎		👍
			Performance consistency			👍	
4	Cache performance	CacheBench	*Read*	Cache bandwidth	👎		👍
			Write		👎	👍	
			Read/Modify/Write		👎		👍
			Performance consistency			👍	
5	File system I/O performance	IOzone	*Disk read*	Disk bandwidth	👍	👎	
			Disk write		👍	👎	
			Performance consistency				👍
6	Application performance	LZMA	File compression	Time	👎		👍
		Sqlite	Database operations	Transactions/s		👎	👍
			Performance consistency				👍

Note: 👍 represents the highest rank in performance comparison; 👎 represents the lowest rank in performance comparison.

the performance gear shifts drastically and UEC exhibits very poor performance.

On the other hand, during cache performance experiments, we found *cache write* operation much faster (more bandwidth) than *cache read* operation. Thus, in any experiment, the ranking of the three CCSPs for overall cache performance can be determined by the ratio of number of *cache read* operations to *cache write* operations. If the number of the *cache read* operations is greater than that of the *cache write* operations (as in our experiments for *cache read/modify/write* operation), PVE outperforms the other two CCSPs (i.e., PVE exhibits the highest bandwidth for *cache read* operation, see Fig.9); otherwise, XCP exhibits a higher bandwidth than the other two CCSPs (because XCP exhibits the highest bandwidth for *cache write* operation, see Fig.9). Among other observations, we found XCP and PVE exhibit more consistent results than UEC. Thus, they are more likely to deliver the expected service level (usually mentioned in service level agreements).

Based on the performance differences of the three CCSPs in our experiments, we observed three performance parameters as critical — for which the performance differences are statistically significant – for the selection of an open source CCSP. These parameters are response efficiency, CPU throughput, and file system I/O performance. For database-oriented applications, the parameter "application performance" also becomes critical. On the basis of our experimental results, we concluded that for CPU intensive applications (that spend more time on CPU than disk read/write operations) as well as for database-oriented applications, PVE is the best choice; for data intensive applications that access data through native file system and have more data read/write operations than CPU operations, UEC is the best choice.

In our current study, we characterized and compared the performance of the three open source CCSPs from the perspective of a common cloud user who wants to build a private cloud. We plan to extend this study with further in-depth experiments to investigate the performance behavior and bottlenecks with these CCSPs. We also intend to extend this study for more CCSPs and for more standard applications. Moreover, we also plan to compare the performance of these CC-

SPs on the cloud resources that are connected through Internet; this study will cover parameters for network performance too. The performance comparison of virtual clusters under major open source CCSPs is also among the future plans.

References

[1] Vaquero L M, Rodero-Merino L, Caceres J, Lindner M. A break in the Clouds: Towards a cloud definition. *ACM SIGCOMM Comput. Commun. Rev.*, 2009, 39(1): 50–55.

[2] Shafer J. I/O virtualization bottlenecks in cloud computing today. In *Proc. the 2nd Workshop on I/O Virtualization*, March 2010, pp.5–12. http://li46–224.members.linode.com/publications/papers/shafer–wiov2010.pdf, Dec. 2014.

[3] Wardley S, Goyer E, Barcet N. Ubuntu enterprise Cloud architecture. Technical White Paper, Ubuntu, 2009. http://www.ubuntu.com/sites/default/files/active/White-%20Paper%20Ubuntu%20Enterprise%20Cloud%20Architecture%20v1.pdf, Apr. 2015.

[4] Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D. Eucalyptus: An open-source cloud computing infrastructure. *Journal of Physics: Conference Series*, 2009, 180(1): 012051.

[5] Saavedra R H, Smith A J. Analysis of benchmark characteristics and benchmark performance prediction. *ACM Trans. Comput. Syst.*, 1996, 14(4): 344–384.

[6] Pu X, Liu L, Mei Y, Sivathanu S, Koh Y, Pu C. Understanding performance interference of I/O workload in virtualized cloud environments. In *Proc. the 3rd IEEE CLOUD*, July 2010, pp.51–58.

[7] Mucci P J, London K, Thurman J. The CacheBench report. Technical Report, ut-cs-98-394, University of Tennessee, 1998. http://icl.cs.utk.edu/projects/llcbench/cachebench.pdf, Jan. 2015.

[8] SchüLLER F. Grid computing with and standard test cases for a meteorological limited area model [Master Thesis]. Institute of Meteorology and Geophysics, University of Innsbruck, 2007. http://imgi.uibk.ac.at/file/157/download?token=KQL58LB2, Jan. 2015.

[9] Deshane T, Shepherd Z, Matthews J, Ben-Yehuda M, Shah A, Rao B. Quantitative comparison of Xen and KVM. In *Proc. Xen Summit*, June 2008. https://www.researchgate.net/publication/228772586Quantitative_comparison_of_Xen_and_KVM/links/004635229734fe42f6000000, Jan. 2015.

[10] Reddy V V, Rajamani L. Evaluation of different hypervisors performance in the private cloud with SIGAR framework. *International Journal of Advanced Computer Science and Applications*, 2014, 5(2): 60–66.

[11] Palankar M R, Iamnitchi A, Ripeanu M, Garfinkel S. Amazon S3 for science grids: A viable solution? In *Proc. the 2008 International Workshop on Data-Aware Distributed Computing*, June 2008, pp.55–64.

[12] Stantchev V. Performance evaluation of cloud computing offerings. In *Proc. the 3rd International Conference on Advanced Engineering Computing and Applications in Sciences*, Oct. 2009, pp.187–192.

[13] Khurshid A, Al-Nayeem A, Gupta I. Performance evaluation of the Illinois cloud computing testbed. Technical Report, Department of Computer Science, University of Illinois at Urbana-Champaign, 2009.

[14] Iosup A, Ostermann S, Yigitbasi M N, Prodan R, Fahringer T, Epema D H. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 2011, 22(6): 931–945.

[15] Garfinkel S. An evaluation of Amazon's Grid computing services: EC2, S3 and SQS. Technical Report, TR-08-07, School for Engineering and Applied Sciences, Harvard University, Cambridge, MA, 2007.

[16] Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman B P, Maechling P. Data sharing options for scientific workflows on Amazon EC2. In *Proc. the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2010.

[17] Kobayashi K, Mikami S, Kimura H, Tatebe O. The Gfarm file system on compute Clouds. In *Proc. IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, May 2011, pp.1034–1041.

[18] Wang L, Zhan J, Shi W, Liang Y. In Cloud, can scientific communities benefit from the economies of scale? *IEEE Transactions on Parallel and Distributed Systems*, 2012, 23(2): 296–303.

[19] Walker E. Benchmarking Amazon EC2 for high-performance scientific computing. ; *Login*, 2008, 33(5): 18–23.

[20] Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D. A performance analysis of EC2 cloud computing services for scientific computing. In *Proc. the 1st International Conference on Cloud Computing*, Oct. 2009, pp.115–131.

[21] Jackson K, Ramakrishnan L, Muriki K, Canon S, Cholia S, Shalf J, Wasserman H J, Wright N. Performance analysis of high performance computing applications on the Amazon web services cloud. In *Proc. the 2nd IEEE International Conference on Cloud Computing Technology and Science* (*CloudCom*), Nov.30–Dec.3, 2010, pp.159–168.

[22] Ye X, Lv A, Zhao L. Research of high performance computing with clouds. In *Proc. the 3rd International Symposium on Computer Science and Computational Technology*, Aug. 2010, pp.289–293.

[23] Ahuja S P, Man S. The state of high performance computing in the cloud. *Journal of Emerging Trends in Computing and Information Sciences*, 2012, 3(2): 262–266.

[24] Nadeem F, Fahringer T. Optimizing execution time predictions of scientific workflow applications in the Grid through evolutionary programming. *Future Generation Computer Systems*, 2013, 29(4): 926–935.

[25] Gupta A, Milojicic D. Evaluation of HPC applications on cloud. In *Proc. the 6th Open Cirrus Summit* (*OCS*), Oct. 2011, pp.22–26.

[26] Juve G, Deelman E, Berriman G, Berman B, Maechling P. An evaluation of the cost and performance of scientific workflows on Amazon EC2. *Journal of Grid Computing*, 2012, 10(1): 5–21.

[27] Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman B, Maechling P. Scientific workflow applications on Amazon EC2. In *Proc. the 5th IEEE International Conference on E-Science Workshops*, Dec. 2009, pp.59–66.

[28] Vecchiola C, Pandey S, Buyya R. High performance cloud computing: A view of scientific applications. In *Proc. the 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, Dec. 2009, pp.4–16.

[29] Evangelinos C, Hill C N. Cloud computing for parallel scientific HPC applications: Feasibility of running coupled atmosphereocean climate models on Amazon's EC2. In *Proc. the 1st CCA*, Oct. 2008.

[30] Li A, Yang X, Kandula S, Zhang M. CloudCmp: Comparing public cloud providers. In *Proc. the 10th ACM SIGCOMM Conference on Internet Measurement*, Nov. 2010, pp.1–14.

[31] Ward J S. A performance comparison of Clouds: Amazon EC2 and Ubuntu enterprise Cloud. Technical Report, Cloud Computing Co-laboratory. University of St Andrews, SICSA DemoFEST, 2009.

[32] Tudoran R, Costan A, Antoniu G *et al.* A performance evaluation of Azure and Nimbus clouds for scientific applications. In *Proc. the 2nd International Workshop on Cloud Computing Platforms*, Apr. 2012, Article No. 4.

[33] Voras I, Orlić M, Mihaljević B. An early comparison of commercial and open-source cloud platforms for scientific environments. In *Proc. the 6th KES International Conference on Agent and MultiAgent Systems: Technologies and Applications*, June 2012, pp.164–173.

[34] Popović O, Jovanović Z, Jovanović N, Popović R. A comparison and security analysis of the cloud computing software platforms. In *Proc. the 10th International Conference on Telecommunication in Modern Satellite Cable and Broadcasting Services* (*TELSIKS*), Vol.2, Oct. 2011, pp.632–634.

[35] Garg S K, Versteeg S, Buyya R. A framework for ranking of Cloud computing services. *Future Generation Computer Systems*, 2013, 29(4): 1012–1023.

Farrukh Nadeem received his Ph.D. degree in computer science in 2009 from the University of Innsbruck, Austria. Currently, he is an assistant professor of King Abdulaziz University, Jeddah. His main research interests include performance modeling and prediction, and scheduling scientific workflows in distributed systems, particularly the Grid and the Cloud. He has been involved in several Austrian and Saudi research and development projects. Farrukh has authored more than 22 papers, including four book chapters.

Rizwan Qaiser received his MPhil degree in computer science in 2012 from National University of Computer & Emerging Sciences, Lahore, Pakistan. Since June 2012, he has been working as a lecturer of computer science. Rizwan's research interests include performance comparison in cloud, virtualization, open source hypervisors, energy efficient cloud computing, cloud security and semantic Web. Apart from academia, Rizwan also works as a freelance software developer.