BUILDING A PRIVATE HPC CLOUD FOR COMPUTE AND DATA-INTENSIVE APPLICATIONS

Moussa Taifi[1], Abdallah Khreishah[2] and Justin Y. Shi[1]

[1]Computer and Information Sciences Department, Temple University, Philadelphia, PA, USA
moussa.taifi@temple.edu,shi@temple.edu
[2]Electrical and Computer Engineering Department, New Jersey Institute of Technology, Newark, NJ, USA
abdallah@adm.njit.edu

ABSTRACT

Traditional HPC (High Performance Computing) clusters are best suited for well-formed calculations. The orderly batch-oriented HPC cluster offers maximal potential for performance per application, but limits resource efficiency and user flexibility. An HPC cloud can host multiple virtual HPC clusters, giving the scientists unprecedented flexibility for research and development. With the proper incentive model, resource efficiency will be automatically maximized. In this context, there are three new challenges. The first is the virtualization overheads. The second is the administrative complexity for scientists to manage the virtual clusters. The third is the programming model. The existing HPC programming models were designed for dedicated homogeneous parallel processors. The HPC cloud is typically heterogeneous and shared. This paper reports on the practice and experiences in building a private HPC cloud using a subset of a traditional HPC cluster. We report our evaluation criteria using Open Source software, and performance studies for compute-intensive and data-intensive applications. We also report the design and implementation of a Puppet-based virtual cluster administration tool called HPCFY. In addition, we show that even if the overhead of virtualization is present, efficient scalability for virtual clusters can be achieved by understanding the effects of virtualization overheads on various types of HPC and Big Data workloads. We aim at providing a detailed experience report to the HPC community, to ease the process of building a private HPC cloud using Open Source software.

KEYWORDS

HPC Cloud, Compute-Intensive Applications, Big Data, Scientific Cluster Operations

1. INTRODUCTION

An HPC cloud is capable of offering HPC-as-a-Service (HPCaaS). Amazon EC2's 42nd entry in the Top 500 fastest computers listing indicates that it is possible for an HPC cloud to deliver a competitive performance given the inherent virtualization overheads. Coupled with potential resource efficiency and flexibility, HPC clouds can dramatically reduce computing costs and improve scientists' productivity. The insatiable resource needs of scientific computing and data-intensive applications further ensure the potential benefits that are critical to the long-term sustainability of the scientific discovery processes. HPC clouds also represent a unique win-win technology/business combination that can maximize resource efficiency, lower computing costs, and ensure a provider's potential profits all at the same time.

DOI : 10.5121/ijccsa.2013.3201

There are three new challenges for using the HPC clouds. The first is the inherent virtualization overhead that directly conflicts with the operational objective of the traditional HPC clusters. Limited by available virtualization technologies, compute-intensive applications that demand very low latency may still have to rely on HPC clusters. Secondly, since the HPC cloud allows the hosting of arbitrary virtual HPC clusters, the scientists have total freedom in choosing and controlling the application stack on every virtual machine. The administrative complexity is challenging for scientists who have already been responsible for domain knowledge and the numerical translation processes. The third challenge is application-programming models. The existing HPC programming models were designed for dedicated homogeneous and non-shared parallel processors. An HPC cloud is typically heterogeneous and more volatile due to resource sharing.

Building a HPC cloud using Open Source software is also a non-trivial challenge. Since the cloud computing support software is relatively young, the availability and maturity of software change rapidly. The differences are subtle, and every decision can either make or break the entire HPC cloud construction process.

This paper reports on the Temple University research team's efforts in building a private HPC cloud alongside of a conventional HPC cluster. The Temple HPC Cloud is a result of negotiations with multiple research groups. We are mindful of multiple conflicting requirements. We report our experiences in managing the conflicts from multiple independent research groups.

This paper is organized as follows:

- Section 2 covers the background of private HPC clouds.
- Section 3 examines three private HPC cloud components: virtualization software, cloud platform, and storage choices. We report our evaluation criteria.
- Section 4 contains a report on automated tools to build and manage virtual clusters. We also describe the design and implementation of HPCFY, a new automation tool for virtual HPC clusters.
- Section 5 contains three performance evaluation reports:
    - Evaluation of VM startup delays.
    - NAS MPI benchmarks for compute-intensive applications.
    - Hadoop performance evaluation for data-intensive applications.
- Section 6 contains the conclusion, and future research directions.

2. BACKGROUND

2.1. Private and Hybrid Computing Clouds

User-centered resource control is the primary motivation for building an HPC cloud. Mitigating factors include either funding source limitations or data security concerns. Being hosted on-site, a private HPC cloud can provide direct user control over every detail of the deployment, management, and customized usage metering of the cloud. Sensitive data faces less threat of being compromised as in public clouds. Internally, application installation and access security can be provided by individual users while leveraging the same resource pool. This allows the scientists to experiment with the latest Open Source software and encourage faster improvements to the collective developments. It also makes economic sense for an enterprise to build its own private cloud using existing resources to reduce administrative costs and to improve resource efficiency.

Since scientific computing applications have insatiable resource needs, it is likely that some applications would grow bigger than a local private HPC cloud could support; hybrid clouds then become desirable.

2.2. Software Availability

Building a private cloud using open source software is a non-trivial trial-and-error process. Customization is a labor-intensive process where care must be taken at every decision point to determine the potential usefulness, support problems, and unintended consequences. Having the experience of building a regular cluster is definitely a plus. Finding help in configuring a particular piece of software can sometimes be a daunting task. Like all other Open Source software, the Open Source cloud computing technologies are continually evolving.

2.3. Hardware Views

Traditional HPC clusters usually follow a simple architecture such as the Beowulf cluster [1], where a head node (Server), an interconnect, and a set of compute nodes constitute the entire service provisioning platform. The head node is used for cluster administration, management, and login. The interconnect provides a way for the nodes to communicate with each other.

As the applications evolve, the HPC clusters have to grow in size to meet the demands. This increases the complexity of the architecture. Today, HPC clusters are typically composed of multiple management nodes, multiple file system nodes, and massively many computing nodes.

The management nodes are composed of a login node plus DHCP (Dynamic Host Control Protocol) and DNS (Domain Name Service) servers. They can also include data access control and user monitoring functions. As the number of computing nodes increases, the pressure increases on the interconnection network and ultimately on the file system for saving the computation results. The file system nodes have seen a steady increase. Large clusters strive to separate file system traffic from computational traffic. Computing nodes typically do not have a large storage capacity. They rely on the centralized parallel file system for data storage and retrieval. Interconnects are also differentiated, such that file system nodes are connected via networks that are capable of higher aggregate bandwidth (less expensive, such as GigaE Ethernet[2]) while computing nodes are typically connected via low latency and high bandwidth networks (more expensive, such as Infiniband [3]).

A private HPC cloud should be able to provision multiple on-demand virtual HPC clusters, i.e., HPCaaS, while sharing a centrally managed pool of HPC resources. An example architecture is shown in Figure 1.

The requirement to support multiple virtual HPC cluster adds a lot of complexity to the cloud software stack. Unlike the traditional HPC cluster, where the resource capacities are fixed by the attached hardware, virtualization allows a new dimension of flexibility that is nonexistent in traditional HPC clusters.

In an HPC cloud, every computing node will run a hypervisor and participate with a user-specified capacity in a full-fledged virtual cluster. Each such cluster should be able to configure its own attached storage and networks. Unlike typical enterprise IT clouds, these virtual HPC clusters must deliver comparable HPC performances using the HPC computing nodes and multiple low latency networks. Unlike enterprise IT clouds, elastic-provisioning of an HPC cloud requires a balance between computation and communication among multiple virtual clusters.

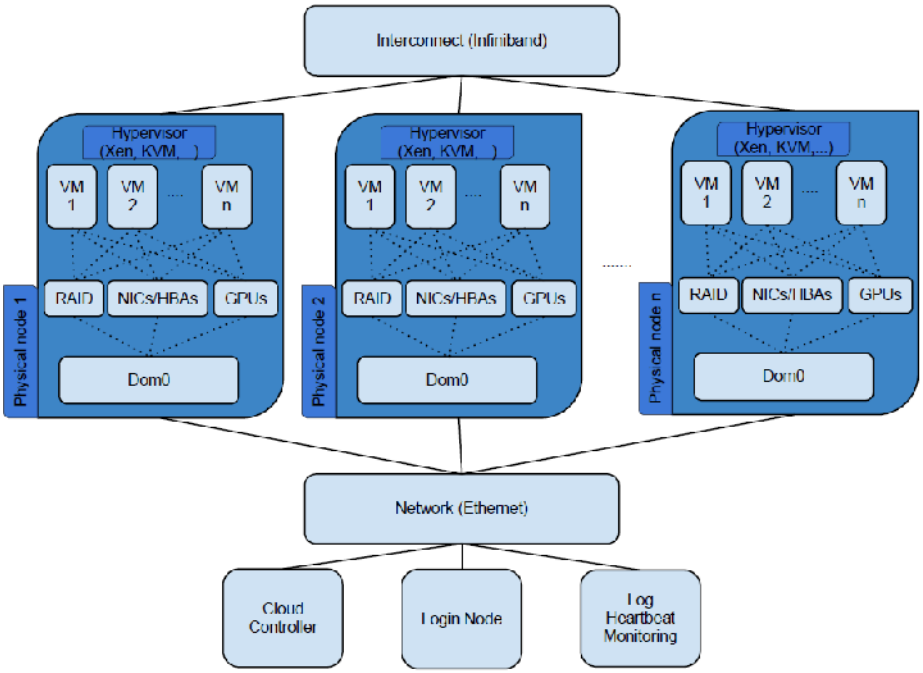

Figure 1: An example of aHPCaaS architecture.

Each virtual cluster can be further customized for specific application types, such as compute-intensive (MPI or OpenMP), data-intensive (big data analytics), or interactive use of mathematical packages, such as Matlab. Most traditional HPC clusters only support batch mode parallel processing. A private HPC cloud seems to be the only solution to meet the diverse application needs from multiple research groups.

3. HPC CLOUD SOFTWARE CHOICES

3.1 Virtualization of High Performance Systems

Linux is the favored HPC operating system. Almost all entries in the Top 500 fastest computers[4] use different versions of the Linux operating system. Due to its open source nature, it is easier to modify and expand an HPC cluster by adapting to the latest processor and networking technologies by different vendors.

For an HPC cloud, the first crucial decision is the choice of the virtualization hypervisor. Pricing, compatibility with the operating system, and the level of hardware support are crucial elements in the decision process. At the time of this writing, there are many good options, such as Xen [5], KVM [6], Vmware [7], Virtual box [8], and others. Each choice seems suitable for a particular cloud application type. We are interested in the virtualization of leading edge high performance processors and networks, such as Graphical Processing units (GPU) and Infiniband networks. The maturity of the virtualization technology also plays a role in the decision process since support from the open source community would be more readily available compared to the less mature projects. Table 1 shows the capability comparisons of the above-mentioned virtualization technologies.

Table 1: Comparison of some key aspects of virtualization technologies

Virtualization	Full virt	Para virt	License	Archit-ectures	Libvirt support	Infini-band support	GPU support
Xen	Yes	Yes	GPL	i686, x86-64, IA64, PPC	yes	yes	Yes
KVM	Yes	Yes	GPL	i686, x86-64, IA64, PPC, S390	yes	yes	Yes
VMware ESX	Yes	No	proprieta ry	i686, x86-64	yes	yes	Yes
VirtualBox	Yes	No	GPL or proprieta ry	i686, x86-64	no	no	yes

Support for the libvirt [9] technology was considered important in our decision process since it unifies different virtualization technologies under the same API. In our analysis, Xen has good support for most of the HPC demands, but KVM is also a viable option. VMware and VirtualBox were considered too expensive for non-business applications, such as computational research.

3.2. Cloud Platform Choices for HPCaaS

The virtual resource management system represents the ``cloud platform". The choice of a cloud platform decides the resource and user management interfaces and functionalities. It is another crucial piece of HPCaaS infrastructure. Ease of user management, security and community support, maturity, and intercloud operability are the decision factors. There are many cloud platforms available today, such as Eucalyptus [10], Nimbus [11], OpenStack [12], and XCP[13]. Table 2 shows the comparisons of available features for these technologies.

Table 2: Comparison of different cloud platforms.

Cloud platform	Client UI	Intercloud operability	License	Main Virtualization support	Maturity
Eucalyptus	EC2 WS, CLI	EC2	BSD	Xen, KVM	High
Nimbus	EC2 WS, CLI, WSRF	EC2	Apache v2	Xen, KVM	Medium
OpenStack	EC2 WS, CLI	EC2	Apache v2	Xen, KVM, UML, HyperV	Low
XCP	CLI	na	GPL2	Xen	Low

We were especially interested in the inter-cloud operability in order to explore external clouds for offloading bursty work in the local private cloud. We found, however, that the hybrid cloud capability is very limited at the time of this report for all of the tools we have studied. Eucalyptus and Nimbus have some support for cloud interoperability. They provide an API that can link the private cloud with public cloud, such as Amazon EC2 HPC instances. XCP only focuses on supporting the Xen platforms. We have also found efforts to extend the functionality of virtualization technology to allow for more flexible cross-platform resources. OpenStack[12] is

such an open source cloud computing project. It is encouraging to observe the emerging competitions in this field when the established entities, such as Eucalyptus and Nimbus, strive to succeed in both open source and commercial spaces. The race is on. A recent survey by [14] presents interesting perspectives in this space. In our evaluation, we choose the Eucalyptus cloud platform.

3.3. Cloud Storage Provisioning

One of the most difficult challenges for building a cloud is its storage provisioning system. In Eucalyptus, the virtual machine's image management is done by the Walrus storage controller. The Walrus service handles the storage management functions of the virtual machine images.
Cloud storage differs from traditional cluster storage in its refined provisioning rules. These include disk image lifecycle management and on-demand storage functionality. Unlike traditional HPC cluster storage devices, where the administrator only installs and configures the file system once, the cloud storage adds an additional conceptual layer. The virtual disks have their own commands for allocation/partition/format/mount/unmount operations for the requesting user. In this sense, without system administration tools or experiences, the complexity for using the HPC cloud is actually higher than the traditional HPC clusters, since the normal administrative routines are all exposed explicitly to the end user.

The complexity of the cloud storage layer also presents itself as delays in the virtual machine (instance) start up time, since the virtual machine images must be transmitted to the node controllers for the entire lifetime of the user's contract with the system. Walrus is the single point of contention. The delay is expected to increase linearly as the number of starting instances increases. The pWalrus project [15] is a recent effort to make Walrus run faster using parallel processing.

Another challenge for the cloud storage controller is the provisioning of storage buckets or on-demand elastic block storage (EBS). Currently, Walrus supports AoE (ATA over Ethernet) and iSCSI (internet SCSI) technologies [16] to provide transparent storage accesses. AoE is suitable for environments where all of the storage nodes are located in the same broadcast domain. iSCSI can be used across multiple broadcast domains [10].

We believe that it is more appropriate to use a distributed/parallel file system to provide a unified namespace for data accesses within the cloud. A scalable distributed/parallel file system has been an on-going research topic in systems area. Many recent projects have shown reasonable advances. These include (not exhaustive) PVFS (Parallel Virtual File System)/OrangeFS [17], HDFS (Hadoop File System)[18], Lustre [19], and GlusterFS [20]. Each of these systems is suitable for a subset of applications by sacrificing properties that are less important for these applications. Table 3 shows the comparison of these file systems. It also shows the design goals and tuning potential for each file system type.

From Table 3, we can see that choosing the best distributed/parallel file system is a non-trivial task. None of the existing file systems seems capable of supporting all application requirements. Limited by the existing technologies, cloud storage system must compromise between availability and performance.

High-end parallel file systems, such as PVFS (Parallel Virtual File System) and Lustre, are popular choices for HPC applications. They both rely on a dedicated storage-area network (SAN) to provide the reliability for large-scale applications. For HPC applications, a distributed SAN is an expensive proposition.

Table 3: Comparison of different distributed/parallel filesystems

Distributed filesystem	Posix compliance	Data Replication	Concurrent write support	Open Source	Performance for replicated data	Maturity
PVFS	Yes	No	Yes	Yes	na	Good
HDFS	No	Yes	No	Yes	Good (no concurrent writes)	Good
Lustre	Yes	No	Yes	Yes	na	Good
GlusterFS	Yes	Yes	Yes	Yes	Medium	Good

When SAN is not appropriate, HDFS (Hadoop File System) can provide a reasonably reliable large-scale storage infrastructure by using replicated data on multiple data nodes. However, there are multiple limitations. The namenode is a single point of failure. There are recent projects that attempt to solve this problem with a checkpoint node [21]or by replicating the namenode metadata using DRDB [22]. Even if the reliability issue can be addressed, HDFS is still not an appropriate HPC cloud storage system choice. HDFS was designed for large-scale data analysis applications. Thus, it allows for ``single write, multiple reads''. Concurrent writes are not supported.

A fourth option is GlusterFS, which is an open source Network Attached Storage (NAS) that specializes in scaling out storage. The main drawback of the GlusterFS is that the replication of the data across pairs of Gluster clients halves the write bandwidth, and thus requires higher aggregate bandwidth. It is also not appropriate for HPC applications. However, the bandwidth drawback would not occur during computation, and thus we consider it a reasonable choice balancing between reliability, performance, and support.

The other options in this space are Redhat's Global File System and the Oracle Distributed File System. These involve proprietary software and require kernel support.

We implemented GlusterFS with Walrus and the storage controller's capabilities to provide EBS storage and virtual machine image management functions.

4. VIRTUAL CLUSTER MANAGEMENT

4.1. Challenges of Virtual Cluster Management

End users have the freedom to build customized HPC clusters using multiple virtual machines, elastic storage, and virtual networks. They also have the task of managing every aspect of these virtual clusters. As discussed earlier, the process requires cluster administration expertise plus the basic cloud resource management skills. These are additional responsibilities on top of being the domain knowledge experts and parallel programmers. The new capability gives greater flexibility to the researchers, such as being ``root'' and install non-standard applications. They also allow customization and optimization of individual applications. However, it is also evident that tools are necessary to ease these administrative chores.

Since typical end users have limited knowledge of cluster administration, many user-oriented tools have emerged to help: the Nimbus Contextualization Tool [23], the Amazon CloudFormation [24], and the MIT Starcluster project [25]. More challenging environments such as auction-based clouds have been examined in the SpotMPI project[36].

The Contextualization Tool can form an HPC cluster and can hide much of the administrative details from the user for Nimbus clouds.

CloudFormation Users can build a cloud infrastructure based on basic building blocks in the Amazon EC2 cloud. The open source community expects that this tool would soon become available for private Eucalyptus clouds.

Starcluster is a python-based scripting toolkit that was also developed to automate the building, configuring, and management of cloud-based clusters for the Amazon EC2 cloud. It can automate cluster construction to include a shared file system, a batch queuing system, and password-less access to virtual machine instances. In addition, an automatic load balancing feature is also available. It can use the statistics of the batch queuing system to adjust the size of the cluster depending on the incoming computational requests.

Once the virtual cluster is ready, traditional HPC tools can be installed to launch applications on hundreds and thousands of virtual computing nodes in parallel.

There are also open-source tools for automating the administration of large-scale virtual clusters. These include Rocks [26], Puppet[27] and Chief [28].

Rocks is a Linux distribution for large-scale clusters. It contains tools for synchronizing software packages and deploying application images to a large number of computing nodes according to the user's specification. This functionality seems not useful for Eucalyptus or EC2 type clouds since the same image can be used to initiate many virtual machines.

More programmatic approaches are adopted in the Puppet and Chief projects. They are two DevOps APIs for the management and configuration of large-scale traditional HPC clusters. Their support for various types of clouds is under heavy development.

It is evident that the user tools community is very active. While a set of published projects are setting the tone for future developments in terms of resource planning and management [37], since the problem space is so large, we still expect many new approaches to meet the HPC sustainability challenges: performance, reliability, scalability, intercloud operability, and usability of virtual clusters all at the same time.

4.2. HPCFY: Puppet-based Virtual Clustering

For our private HPC cloud project, none of the previously mentioned tools would fit the Eucalyptus platform without heavy modifications. The specificity of each tool renders them useless to us. This has led to the development of a minimalist, but extendable, clustering tool (HPCFY) that is based on the configuration tool called Puppet [27].

Figure 2 shows an overview of the HPCFY clustering process to provide dynamic contextualization for a group of independent virtual machines. HPCFY contains a set of Puppet classes and utilities that allow easy cluster management [29].

The users login to the head node of the private cloud from which they deploy their own clusters. The user starts by requesting a set of VMs from the cloud controller by specifying a virtual machine image and type as well as the number of instances to be used. The cloud controller looks at the current availability and allocates the requested number of VMs. Subsequently, the user selects one of the nodes to be the head node of the user cluster.

In the current implementation, the head node is assigned as the Puppet Master and is also the NFS server for the user cluster. The user then sends the cluster configuration to the Puppet Master that deploys the cluster configuration to the rest of the nodes. Once the cluster is stabilized, the user will be able to launch parallel applications from the user cluster head node.

The HPCFY project is openly available and can be easily downloaded and extended. Currently, we support popular HPC packages, such as MPI and Hadoop/MapReduce with the large scale data mining package Mahout. It also provides automatic configuration of a NFS-based file system, distributed user accounts security configurations, and cluster monitoring using the Ganglia project [30].

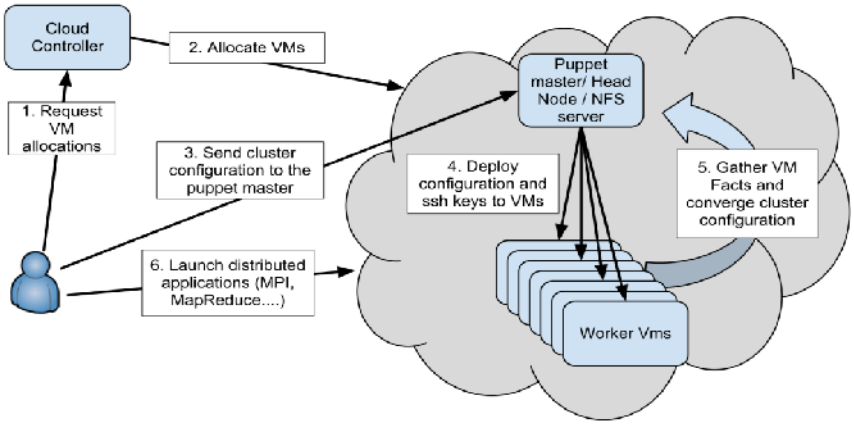

Figure 2: Overview of the clustering workflow using HPCFY.

The open and minimalist nature of the HPCFY/Puppet framework allows the users to easily customize cluster configuration to fit their specific software and hardware needs. This can be done with little interaction with the cloud administrator utilizing the complete access to the allocated virtual machines. We expect that this new flexibility could lead to better time-to-solution for many different research groups.

The next section reports on the computational results using the virtual clusters built with the HPCFY framework.

5. PERFORMANCE RESULTS

This section reports on the computational results from benchmarking the private cloud. We measure the virtual cluster startup time and the performance scalability for compute-intensive and data-intensive applications. We compare the performance of physical and virtual processing cores. The objective is to expose the virtualization performance overhead and to identify potentials for future improvements.

We aim to gain an understanding of the performance characteristics of the experimental private HPC cloud. This section is organized as follows:

- Experiment setup.
- Evaluation of VM startup delays.
- Performance of compute-intensive applications:
    - Performance Comparisons of Physical and Virtual Cores
    - Computational scalability analysis of virtual clusters.

- Performance of data-intensive applications:
    - Performance analysis of the effect of VM types on Hadoop applications.

5.1. Experiment Setup

We planned four experiments using our private HPC cloud [31]. It has 12 physical nodes in two groups: 8 regular multicore nodes and 4 multicore with GPU nodes. The regular multicore node is a 2 x Intel Xeon CPU X5660 processor with a total of 12 cores, 12G RAM, and a 1 Gigabit Ethernet card. Each multicore GPU node has a 2 x Intel Xeon CPU E5630 with a total of 16 cores, 24G RAM, and 2 Nvidia c2050 cards. Every node has a 1 Gigabit Ethernet card and an Infiniband QDR interface. The operating system is a modified version of Debian 6.0 Squeeze with Xen 4.0 as the hypervisor. One of the 12 nodes is the dedicated head/login node where no virtual machine is allowed. The same node hosts the cloud system management components, (i.e., Eucalyptus cloud controller, walrus service, storage controller, and cluster controller).

Firstly, we measure the virtual cluster startup time and increase the number of simultaneous VMs and varying VM types.

In the second exercise, we test the compute-intensive NAS NPB and NAS NPB-MZ benchmarks[32] to evaluate the computational capabilities of the virtual cluster. We use the HPCFY tool to help with test automation. We record the effects of different data sizes by running classes C and D of the NAS benchmarks and the pseudo applications part of the benchmark. We also tested the multi-zone benchmark that relies on a hybrid MPI-OpenMP programming model.

The final exercise concerns data-intensive applications. We run four data-intensive benchmark applications: DFSIO, Sort, Wordcount, and Kmeans clustering, using the Hadoop/MapReduce and Mahout packages.

5.1. Virtual machine startup times

The virtual machine (VM) start up time depends on the size of the VM image, the bandwidth available, and the requested VM type.

Varying the number of simultaneous VMs tests the scalability of the cloud management system as the walrus and the storage controllers responds to the VM requests. It directly stresses the network as it must determine the data transfer rates from the location of the VM image to the node controller. To complete the virtual machines, the node controller needs to create dedicated disk space for the virtual machine by attaching the elastic storage specified by the user. Once complete, this non-trivial process allows the node controller to boot the virtual machine.
Table 4 describes the VM types used in the experiments. The different types serve different purposes. Smaller instances can be used for development and testing while the larger instances can be used for production work. Each VM type will have a different startup time.

Table 4. VM types used in the experiments.

Name	CPUs	Memory(MB)	Disk(GB)
m1.small	1	128	2
c1.medium	1	256	5
m1.large	2	512	10
m1.xlarge	2	1024	20
c1.xlarge	4	2048	20

In this experiment, we used a Debian 5.0 image of size 1.2GB. We varied the number of simultaneous instances.

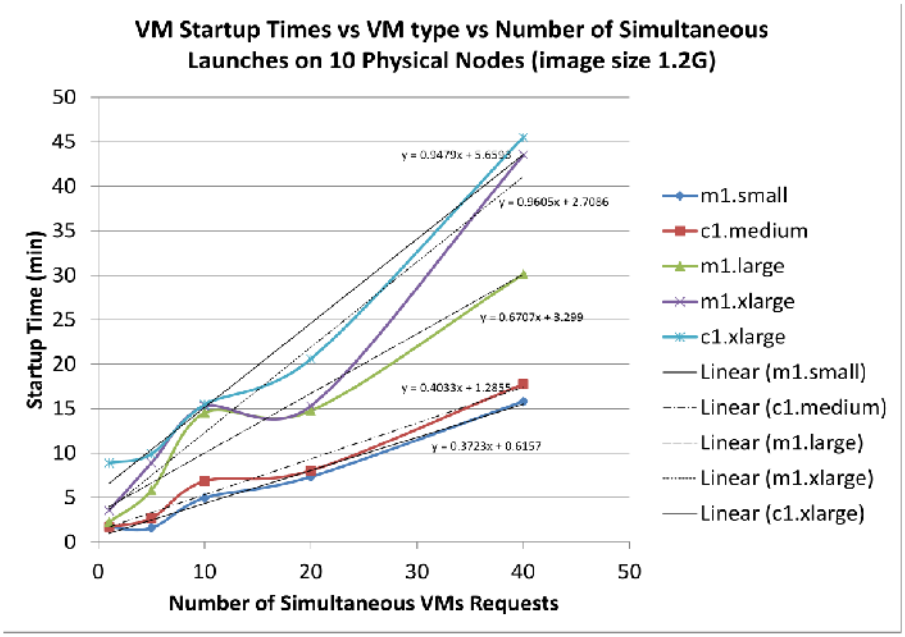

Figure 3: Startup times vsthe number of VMs and VM Types.

Figure 3 shows that the startup time behavior can separated into 3 groups. We use a linear fitting to show the similarity of the three groups. The elastic user storage size affects the startup time greatly. In comparison, different CPU and RAM settings have negligible effects. The startup time of smaller instances (m1.small and c1.medium, 2 and 5 GB respectively) increase by a factor of 0.3 to 0.4 when increasing the number of requested VMs. The third largest VM type's (m1.large, 10GB) startup time increases by a factor of 0.6. The two largest VM types (m1.xlarge and c1.xlarge, 20 GB) show almost linear time increase with a factor of 0.9.

From this figure we can also see that the initial operating system image's size is the same of all the tests. This means that the transfer time of the image is the same of all the tests. In fact, we noticed that the longest time spent is in the creation of the root device of the virtual machine on the node controller physical storage. As a best practice, we currently encourage users to keep the image size as small as possible to reduce the startup time and use external storage such as EBS to have access to larger storage. This effectively decouples the startup times from the size of the operating system images.

These results can be useful when making scheduling decisions. Large initial waiting times are to be expected. The startup time will lengthen when there are concurrently running VMs. For users of a batch scheduling system that provisions virtual clusters, the queue waiting time should include the startup time in addition to the dynamic scheduling and back-filling overheads/waiting times.

5.2. Compute-Intensive Benchmarks

5.2.1.Performance Comparison of Physical and Virtual Cores

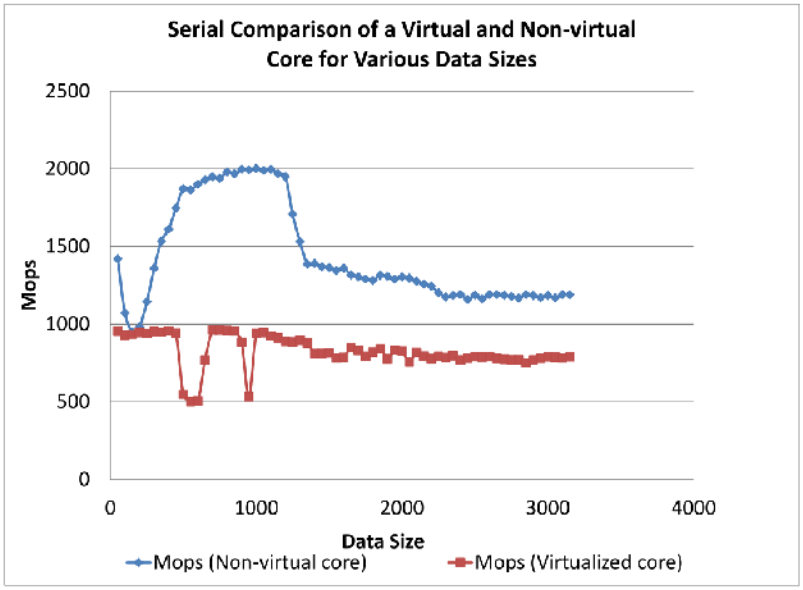

Figure 4: Performances of a virtual and physical core for matrix multiplication.

We evaluated the virtualization overhead by measuring the serial processing capability difference using a single virtualized core versus a physical core. Figure 4 shows the Mops (Million Operations Per Second) of the two configurations. The test program is a matrix multiplication application. We vary the size of the matrices. We notice that the physical core's performance has a caching effect. The caching effect is non-existent to the virtual core. The actual virtualization overheads can amount to 40% to 50 % compared to a physical core on the same machine.

5.2.2. Computational Scalability of Virtual Clusters

In this section, we report on two sets of experiments that evaluates the computational scalability of the virtual machines. The tests include different problem sizes, different programming models, and granularity for three parallel applications from the NAS benchmarks [32]. The NAS benchmarks are widely used to measure the capabilities of supercomputers.

We used three CFD (Computational Fluid Dynamics) applications: scalar pentadiagonal (SP), block tridiagonal (BT), and lower upper diagonal (LU). These applications have different types of parallelism and computation/communication ratios[32]. We tested two classes, C (small) and D (16x larger than C).

We used the largest VM type. The default configuration of the c1.xlarge (Table 4) includes 4 CPU cores and 2 GB of memory. The memory size is considered too small for this type of application. Depending upon the research groups' needs, the cloud administrator needs decide upon the suitable VM types based on available physical resources.

Figures 5 and 6 show the relative speedups for class C and class D versions of the NPB and NPB-MZ benchmarks. For class C, we used 1 to 16 VMs for NPB and 1 to 25 VMs for NPB-MZ. For

class D, we used 9 to 25 VMs for NPB and 9 to 25 VMs for NPB-MZ. This is because the VMs have limited RAM; it was not possible to fit a class D problem into small VMs.

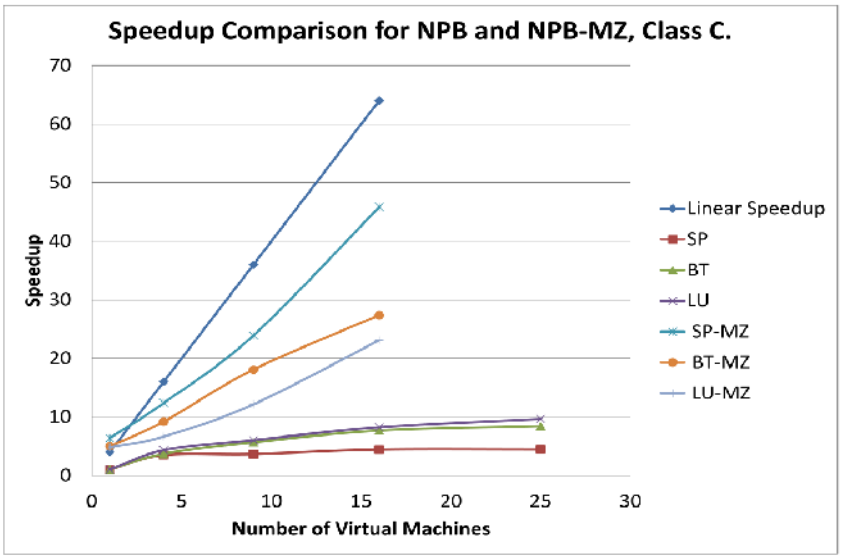

Figure 5: Relative Speedup for NPB and NPB-MZ, (Class C).

For the NPB benchmark, each MPI job requires the explicit definition of how many MPI tasks are to run on each node. In our case, only one MPI task per node was possible before swapping. Using only one core of each virtual machine leads to high network communication between tasks. For the class D benchmarks, the problem sizes are 16x larger; even though the VM memory size could barely hold a single MPI task, the overall performance behavior is similar to the class C benchmarks.

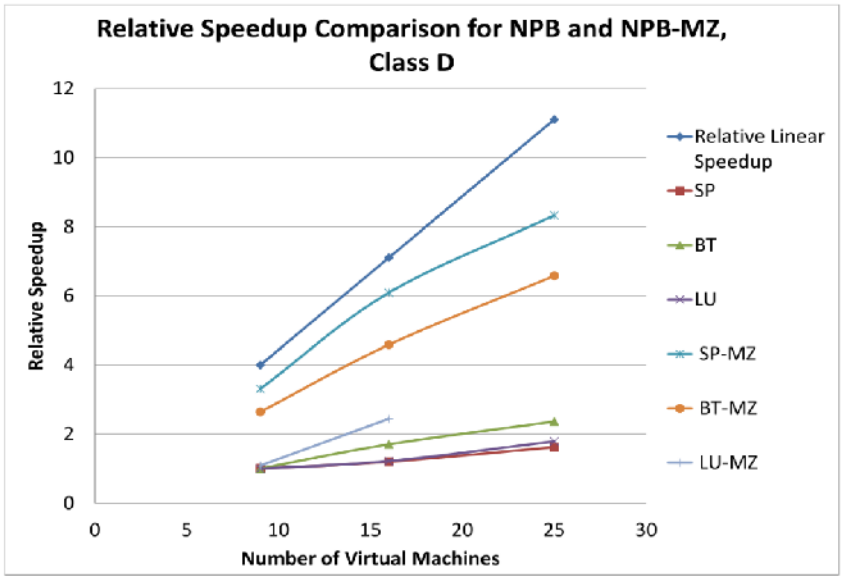

Figure 6: Relative speed up comparison for NPB and NPB-MZ, (Class D).

The multi-zone benchmarks (NPB-MZ) use MPI+OpenMP programming. These results have consistent improvements compared to the MPI-only versions. This is because the MPI+OpenMP version allows efficient use of all cores available within each VM under a small memory footprint. This insight can be useful when using memory-constrained VMs. Another observation is that, while the speedup differences between the three NPB benchmarks are small, the differences are more pronounced for the NPB-MZ benchmarks. One possible reason is the altered computation/communication ratios between the different implementations of the same algorithm.

5.3. DATA-INTENSIVE BENCHMARKS

5.3.1. Traditional Cluster Infrastructures and Data-Intensive Workloads

One of the challenges of a traditional cluster is the support of new research tools. Maintaining the correct version of the software desired by research groups is a non-trivial task for cluster administrators. Since traditional HPC clusters are optimized for compute-intensive applications, data-intensive tasks are deemed inefficient. Often these software requests are denied access to the HPC processors. HPCaaS clouds provide an ideal solution that gives the user the total control over the virtual machines. Although the virtual machines are slower than the traditional cluster, and the researchers are facing the complexities of virtual cluster administration, all of the overhead maybe insignificant in view of scientific discovery processes in the long run. We actually expect this newly found freedom to improve the speed of scientific discoveries.

One of the recent tools for big data research is the use of ``map-reduce" framework, that has been popularized by Google [33] and picked up by many research groups and commercial companies. The big data research groups use web-scale social media data in combination with scientific instrumentation data to solve fundamental problems from gene sequencing, and protein folding, to social, economic and political trend research.

In this section, we report on the computational results of data-intensive benchmarks. Our goal is to understand the performance effects of using the virtual machines for a Hadoop cluster. We study the performance effects of different VM types and workload types.

5.3.2. HADOOPMAPREDUCE BENCHMARKS

In this section, we present the performance results using virtual Hadoop clusters. We used two VM types, c1.xlarge and m1.xlarge. The configurations of these two VM types are summarized in Table 5.

Table 4: Configuration of the VMs used for the Hadoop Clusters.

VM type	Configuration element	Description
c1.xlarge	Number of virtual cores	4
	Memory Size	4GB
m1.xlarge	Number of virtual cores	2
	Memory Size	2GB
c1.xlarge and m1.xlarge	Hard Drive	20GB
	Network	1 Gigabit Ethernet NIC
	Operating system	Ubuntu 10.04 Lucid, 2.6.27.21-0.1-xen
	JVM	1.7
	Hadoop version	Hadoop 0.20.2-cdh3u3

We ran 4 classes of Map Reduce benchmarks applications: DFSIO, Sorting, Wordcount, and Kmeans. DFSIO tests the IO performance of the virtual HDFS. We measure the read and write performances. The sorting benchmark is a micro-version of the popular Terasort benchmark[34]. The Wordcount benchmark consists of aggregating the occurrence of individual words in a set of files residing on the HDFS storage. The kmeans benchmark uses the Mahout library [35] to cluster data into a number of related groups. The last three applications should show some level of speedup.

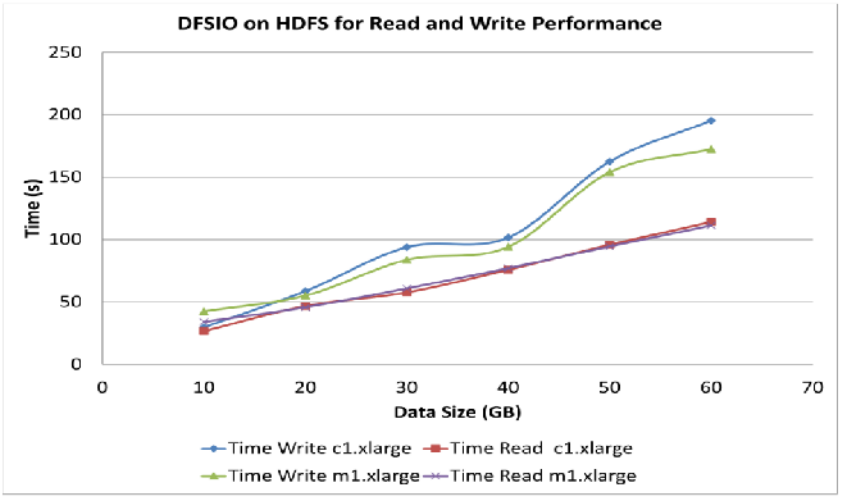

Figure 7: DFSIO performance comparison

We are interested in the performance effects by different VM types. We used HPCFY to start a cluster using one type of VM, c1.xlarge or m1.xlarge. We record the performance of each configuration. The two clusters differ only in terms of core count per VM(4 vs 2) and available Memory (4GB vs 2GB). Each virtual Hadoop cluster consists of 10 VMs interconnected with a virtual Gigabit Ethernet network. To control the environment, we used a fixed 20GB storage per VM. The physical storage is composed of 5x500GB disks in a RAID5 configuration, and the Xen hypervisor provides the access for each VM. This is not an optimal set up, because Hadoop's best practices advises to have a looser JBOD setup for the hard disks since the HDFS is supposed to provide redundancy at the software level. This should not affect the performance comparison for the present experiments. In our experiments, we made sure that only one VM is running on each physical machine to circumvent the limitations of our current storage configuration.

Figure 7 reports the DFSIO results. We use the HDFS system to read and write 10 files with sizes ranging from 10 to 60 GB. As expected, reads are faster than writes. Changing the VM type does not affect the IO performance much.

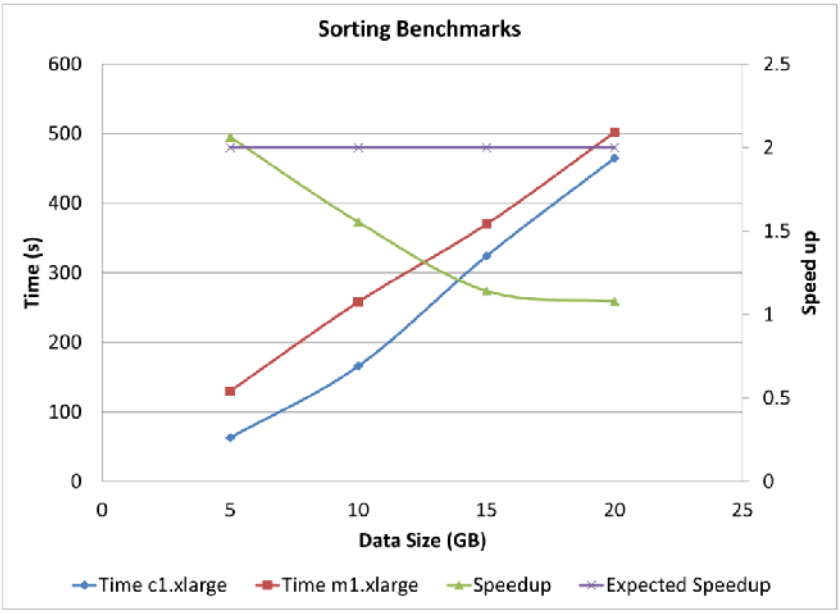

Figure 8: Terasort performance comparison

Figure 8 reports the performance of the sorting benchmark, Terasort, using VMs with different core-counts: c1.xlarge(4 cores) and m1.xlarge(2 cores). Using the same number of VMs, the expected speedup is 2x for c1.xlarge clusters. The recorded performances are for two different virtual clusters. The c1.xlarge virtual cluster performed consistently better. The benefits of multicore processing decreases as the problem size increases until 15 GB. Since the computing complexity$O(nlg(n))$ is higher than the communication complexity$O(n)$, the speed differences stabilize.

Figure 9: Wordcount performance comparison.

Figure 10 reports the virtual cluster performance for the Wordcount application with increasing sizes of text corpus. Unlike the sorting benchmark, the c1.xlarge cluster could only delivers around 1.5x speedup. This is because that the computing and communication complexities are both linear, $O(n)$.The multicore benefits diminish after the network is saturated.

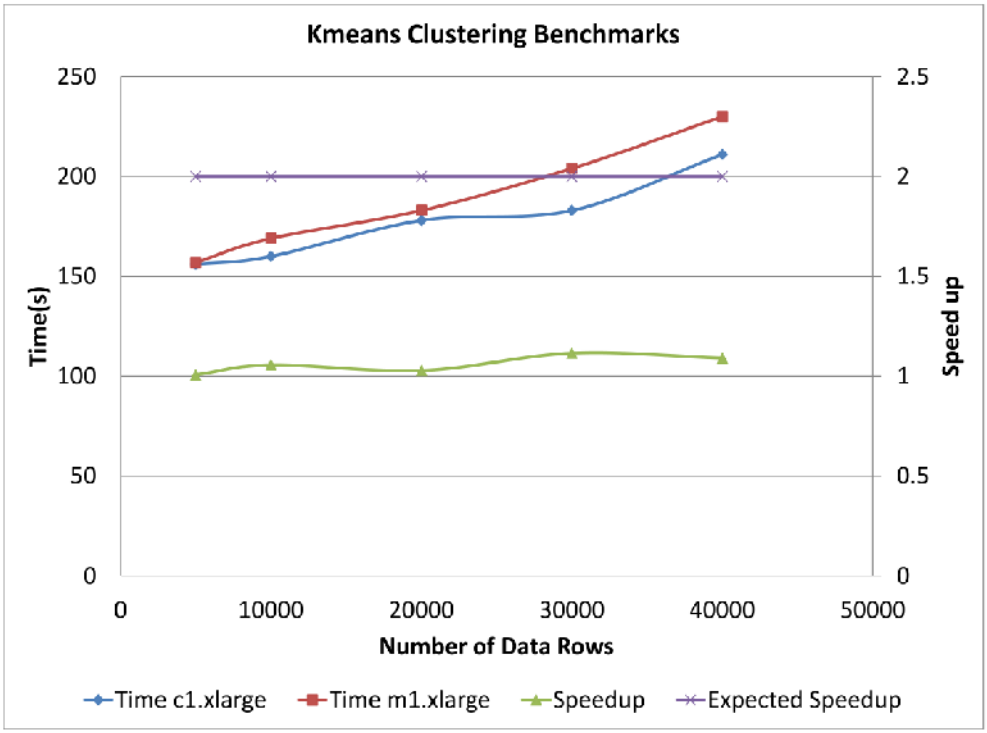

Figure 10: Kmeans performance comparison

For the Kmeans clustering application, we used the Open Source project Mahout [35]to perform clustering on a range of data points. Each data point has 60 dimensions. The distance measure is the Euclidian distance between any two data points. We set the number of clusters to 10. The performance results show the time spent during 10 iterations of the program. The Kmeans benchmark, Figure 10, reports speedup around 1 with increasing data sizes. This is due to the iterative nature of the Kmeans algorithm, where each iteration is processed in Mahout/Hadoop as a related but separate job. This incurs heavy overhead because of the frequent starting and stopping of Hadoop jobs. This effect, in addition to the network bottleneck at larger data sizes, contributes to the unchanged speedup when using larger VMs. Applications with frequent synchronization barriers do not benefit from multicore technologies.

3. CONCLUSION AND FUTURE DIRECTIONS

Cloud-based HPC computing is a fast-evolving research and development area. Although there are many difficulties, we expect that these difficulties will be overcome in the near future. Cloud computing has already changed the landscape of IT service industries. The dramatically improved resource utilization benefits are catching people's attention. Theoretically, HPC applications can amplify these benefits due to their insatiable resource needs. In this paper, we report on our practice and experiences in building a private HPC cloud from scratch. We share our results, the tools, and the technologies that we have evaluated and used. Our experimental private HPC cloud

is currently in production at Temple University. We serve research groups in computer science, mathematics, physics, chemistry, biology, engineering, earth sciences, and medical sciences.

There are still major weaknesses in the existing cloud controllers. The primary challenge is to gain performance and reliability at the same time as we scale-up the HPC cloud. This challenge is particularly critical to the cloud storage for its important role in the entire system. There is also much work to do in the virtual cluster controller for end users. For broader impacts and higher productivity for HPC research and education, the tools must come through in a timely manner.
By taking advantage of the Puppet project, we introduced a user cluster management tool, HPCFY, for building virtual clusters. We also presented benchmark results for compute-intensive and data-intensive applications on the virtual machine clusters.

As the advances in computing hardware and networking technologies are slowing, the next revolution would be in the area of HPC clouds, for their applications have growing resource needs. Although it may seem far-fetched at the moment, an ideal would be to deliver higher performance and reliability at the same time as the application acquires more resources. A private HPC cloud would be the perfect testbed for experimental HPC research and development while providing production quality service to the general research communities.

HPCaaS is a very young and challenging field in systems research. It promises to contribute positively to the development of computational sciences in theory and in practice. Our future work will be focused on the sustainability of HPC applications when acquiring more computing and communication resources, especially the challenges of gaining performance, reliability, scalability, intercloud operability, and usability of virtual clusters all at the same time. We will explore auction-based HPC clouds in order to test the application's ultimate robustness under extreme conditions. We believe that an extreme-scale HPC application is only possible if it can sustain extreme volatile conditions on a large scale.

ACKNOWLEDGEMENTS

The authors would like Thomas Stauffer, AakashPradeep and Dr.Jie Wu for useful feedback and suggestions, as well operational support during the design and deployment phases of the private cloud described in this paper.

REFERENCES

[1] T. Sterling, Beowulf cluster computing with Linux, The MIT Press, 2002.
[2] R. Seifert, Gigabit Ethernet: technology and applications for high speed LANs, Addison-Wesley Reading, Massachusetts, 1998.
[3] I. T. Association, InfiniBand Architecture Specification: Release 1.0, InfiniBand Trade Association, 2000.
[4] Top500 Supercomputing Site [Website], February 5 (2011).
[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, in: Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM, pp. 164–177.
[6] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori, in: Proceedings of the Linux Symposium, volume 1, pp. 225–230.
[7] M. Rosenblum, in: Proceedings of Hot Chips, pp. 185–196.
[8] V. Oracle, Virtualbox, user manual, 2011, 2011.
[9] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehorster, A. Brinkmann, in: Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE), pp. 574–579.
[10] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, in: Proc. 9th IEEE/ACM Int. Symp. Cluster Computing and the Grid CCGRID ’09, pp. 124–131.

[11] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, M. Tsugawa, Cloud computing and applications 2008 (2008).
[12] R. Clark, in: AGU Fall Meeting Abstracts, volume 1, p. 01.23
[13] C. Systems, Xen cloud platform - advanced virtualization infrastructurefor the clouds, 2010.
[14] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, Q. Li, in: InformationScience and Engineering (ISISE), 2009 Second International Symposiumon, Ieee, pp. 23–27.
[15] Y. Abe, G. Gibson, in: Cluster Computing Workshops and Posters(CLUSTER WORKSHOPS), 2010 IEEE International Conference on,IEEE, pp. 1–7.
[16] X. Gao, M. Lowe, Y. Ma, M. Pierce, in: E-Science Workshops, 2009 5thIEEE International Conference on, IEEE, pp. 71–78.
[17] P. Carns, W. Ligon III, R. Ross, R. Thakur, in: Proceedings of the 4thannual Linux Showcase & Conference-Volume 4, USENIX Association,pp. 28–28.
[18] D. Borthakur, Hadoop Project Website (2007).
[19] P. Braam, R. Zahir, Cluster File Systems, Inc (2002).
[20] Y. Lokeshwari, B. Prabavathy, C. Babu, in: International Conferenceon Emerging Technology Trends (ICETT), volume 2, p. 1.
[21] D. Borthakur, Hdfs architecture guide, 2008.
[22] S. Son, R. Beckinger, D. Baker, Journal of Systems and Software 42(1998) 193–204.
[23] K. Keahey, T. Freeman, in: eScience, 2008. eScience'08. IEEE FourthInternational Conference on, IEEE, pp. 301–308.
[24] Amazon web services, http://aws.amazon.com/ec2, 2011.
[25] V. Fusaro, P. Patil, E. Gafni, D. Wall, P. Tonellato, PLoS computationalbiology 7 (2011) e1002147.
[26] P. Papadopoulos, M. Katz, G. Bruno, Concurrency and Computation:Practice and Experience 15 (2003) 707–725.
[27] J. Turnbull, Pulling strings with puppet: configuration managementmade easy, Springer, 2008.
[28] T. Delaet, W. Joosen, B. Vanbrabant, in: Proceedings of the 24th LargeInstallations Systems Administration (LISA) conference, San Jose, CA,USA, volume 11, p. 2010.
[29] M.Taifi,Hpcfy-virtualhpcclusterorchestrationlibrary,https://github.com/moutai/hpcfy, 2012.
[30] M. Massie, B. Chun, D. Culler, Parallel Computing 30 (2004) 817–840.
[31] M.Taifi,J.Shi,Tcloud,hpcaasprivatecloud,http://tec.hpc.temple.edu, 2012.
[32] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum,R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, et al., Journal ofSupercomputer Applications 5 (1991) 63–73.
[33] J. Dean, S. Ghemawat, Communications of the ACM 51 (2008) 107–113.
[34] O. OMalley, A. Murthy, Proceedings of sort benchmark (2009).
[35] S. Owen, R. Anil, T. Dunning, E. Friedman, Mahout in action, ManningPublications Co., 2011.
[36] J.Y.Shi, M.Taifi, A.Khreishah,"Resource Planning for ParallelProcessing in the Cloud," in IEEE 13th International Conferenceon High Performance and Computing, Nov. 2011, pp. 828-833.
[37] M.Taifi, Justin Y. Shi,A.Khreishah,Spotmpi: A Framework for Auction-based HPC Computing using Amazon Spot Instances. LNCS, 7017(3):109–120, 2011.

AUTHORS

MoussaTaifi is a PhD candidate at Temple University, PA, in Computer and Information sciences Department. He holds a Master in information technology from Lappeenranta University of technology with a major in communication engineering. He has a minor in marketing and in information processing. In 2005-06, he was a visiting student at Haverford College PA, USA. He received his Bachelors degree of science in general engineering from Al Akhawayn University in Ifrane, Morocco.

His research interests lie in HPC parallel software systems, HPC in the Cloud, Big data architectures andartificialintelligence.

AbdallahKhreishah received his Ph.D andM.S. degrees in Electrical and Computer Engineeringfrom Purdue University in 2010 and2006, respectively. Prior to that, he received hisB.S. degree with honors from Jordan Universityof Science & Technology in 2004. Immediatelyafter finishing his Ph.D, Abdallah joined the Departmentof Computer & Information Sciencesat Temple University as an Assistant Professor.During the last year of his Ph.D, he worked withNEESCOM. In Fall 2012, he joined the ECEdepartment of New Jersey Institute of Technology as an AssistantProfessor. Abdallah is an active researcher. His research spans theareas of network coding, wireless networks, congestion control, cloudcomputing, network security, and database systems for large projects.His research projects are funded by the National Science Foundationand the UAE Research Foundation.

Justin Y. Shi (a.k.a. Yuan Shi) is an Associate Professor and Associate Chairman of the CIS Department at Temple University. Dr. Shi earned his B.S. degree in Computer Engineering from the Shanghai Jiaotong University in Shanghai, China; his M.S. and Ph.D. degrees in Computer Science from the University of Pennsylvania in Philadelphia. His primary research area is sustainable large scale systems. These include high performance computing, very large scale databases, very large scale messaging systems and very large scale storage networks.