Evaluating MapReduce on Virtual Machines: The Hadoop Case*

Shadi Ibrahim[1], Hai Jin[1], Lu Lu[1], Li Qi[2], Song Wu[1], and Xuanhua Shi[1]

[1] Cluster and Grid Computing Lab
Services Computing Technology and System Lab
Huazhong University of Science & Technology, Wuhan, 430074, China
{shadi,hjin}@hust.edu.cn
[2] Operation Center
China Development Bank, Beijing, China
quick.qi@gmail.com

Abstract. MapReduce is emerging as an important programming model for large scale parallel application. Meanwhile, Hadoop is an open source implementation of MapReduce enjoying wide popularity for developing data intensive applications in the cloud. As, in the cloud, the computing unit is virtual machine (VM) based; it is feasible to demonstrate the applicability of MapReduce on virtualized data center. Although the potential for poor performance and heavy load no doubt exists, virtual machines can instead be used to fully utilize the system resources, ease the management of such systems, improve the reliability, and save the power. In this paper, a series of experiments are conducted to measure and analyze the performance of Hadoop on VMs. Our experiments are used as a basis for outlining several issues that will need to be considered when implementing MapReduce to fit completely in the cloud.

Keywords: Cloud Computing, Data Intensive, MapReduce, Hadoop, Distributed File System, Virtual Machine.

1 Introduction

The computing world is undergoing a significant transformation from traditional non-centralized distributed system architecture, typified by distribute data and computation on different geographic areas to a centralized cloud computing architecture, where the computations and data are operated somewhere in the cloud, data centers owned and maintained by third party. However, in term of resources, the three main characteristics of cloud are: (1) On-demand unlimited data storage, (2) on-demand computation power with no lock, mainly represented as VMs, and (3) using internet, limited bandwidth connection, to access, use and process these resources.

* This work is supported by National 973 Key Basic Research Program under grant No.2007CB310900, Information Technology Foundation of MOE and Intel under grant MOE-INTEL-09-03, and National High-Tech 863 R&D Plan of China under grant 2006AA01A115.

M.G. Jaatun, G. Zhao, and C. Rong (Eds.): CloudCom 2009, LNCS 5931, pp. 519–528, 2009.

The new surge and interest of cloud computing in accompanied with exponentially growing of data size generated from digital media (images/audio/video), web authoring, scientific instruments, and physical simulations. Thus, how to effectively process these immense data sets is becoming a challenging issue in the cloud. While, the traditional data intensive system, typified by moving data to computing, design and programming models are, due to the bottleneck of the internet when transferring large amount of data to the computing nodes, to be not efficient for cloud [1]. Data-aware approach is proven to be efficient and robust, where data and computation are collocated. This approach has been widely used and studied, especially after the great success of Google version, namely Google File System (GFS) [2] and MapReduce [3] (e.g. Google uses its MapReduec framework to process 20 petabytes of data per day [3]). Recently, many projects are exploring ways to support MapReduce on various types of distributed architecture (e.g. Hadoop [4] for data intensive applications, Phoenix [5] for multi-core programming), and for wider applications [6, 7].

Hadoop [4] is an open source implementation of MapReduce sponsored by Yahoo. It has been widely used and experienced for large scale data applications in the clouds [6, 7]. Furthermore, Hadoop is advocated by industry's premier web players - Google, Yahoo, Microsoft, and Facebook - as the engine to power the cloud [8]. As in the cloud, the computing unit is mostly VM-based (Amazon Elastic Cloud Computing [9] and GoGrid [10] are providing VM-based computing infrastructure as a service), it is feasible to demonstrate the applicability of MapReduce in virtualized data center. Although the potential for poor performance and heavy load undoubtedly exists, virtual machine can instead be used to help to fully utilize the system resources, ease the management of such systems as well as improve the reliability, and power saving (i.e. virtual machines have been a promising approach for various distributed systems [11-14]). More recently, Amazon added a new service, called Amazon Elastic MapReduce [15], enables customers easily and cost-effectively process vast amounts of data. It utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (EC2) and Simple Storage Service (S3) [16].

To practically introduce the challenges and the opportunities of combining MapReduce and VM technologies, in this paper, a series of experiments are conducted to measure the performance of Hadoop on VMs in different scenarios. First, we comparatively evaluate the performance of Hadoop Distributed File System (HDFS) on both physical and virtual cluster. Then we analyze the performance of the Hadoop MapReduce framework in virtualized cluster. In summary, the main contributions of our work are:

- We are first in the cloud community to carry out detail performance evaluations when deploying Hadoop on virtualized cluster.
- We elaborate several issues that can be used for better fit of MapReduce in the cloud.

The rest of this paper is organized as follows. Section 2 provides the background knowledge related to this work including an overview of the MapReduce programming model and why deploying MapReduce on virtual machines. In section 3, we take an overview on our experimental methodology, platform and benchmarks. We then present our results in section 4. While section 5 discusses some open issues and the lessons learned from our experiments. Finally, we conclude the paper and propose our future work in section 6.

2 Background and Motivations

In this section, we briefly introduce MapReduce model and its widely used implementation, Hadoop. Then we propose some aspects when using VMs with MapReduce.

2.1 MapReduce

MapReduce [3] is a programming model for data intensive computing inspired by the functional programming. It is simply represented in two functions:

- The map function, written by the user, processes a key/value pair to generate a set of intermediate key/value pairs.

$$map\ (key_1,\ value_1) \rightarrow list\ (key_2,\ value_2)$$

- The reduce function, also written by the user, merges all intermediate values associated with the same intermediate key.

$$reduce\ (key2,\ list\ (value2)) \rightarrow list\ (value2)$$

The MapReduce model allows programmers to easily design parallel and distributed applications, simply by writing Map/Reduce components, while the MapReduce runtime is responsible for parallelization, concurrency control and fault tolerance.

2.2 Hadoop

Hadoop [4] is java open source implementation of MapReduce sponsored by Yahoo. The Hadoop project is a collection of various subprojects for reliable, scalable distributed computing [4]. The two fundamental subprojects are the Hadoop MapReduce framework and the HDFS.

HDFS is a distributed file system that provides high throughput access to application data [4]. It is inspired by the GFS. HDFS has master/slave architecture. The master server, called NameNode, splits files into blocks and distributes them across the cluster with replication for fault tolerance. It holds all metadata information about stored files. The HDFS slaves, the actual store of the data blocks called DataNodes, serve read/write requests from clients and propagate replication tasks as directed by the NameNode.

The Hadoop MapReduce is a software framework for distributed processing of large data sets on compute clusters [4]. It runs on top of HDFS. Thus data processing is collocated with data storage. It also has master/slave architecture. The master, called *Job Tracker* (JT), is responsible for : (a) querying the NameNode for the block locations, (b) considering the information retrieved by the NameNode, JT schedules the tasks on the slaves, called *Task Trackers* (TT), and (c) monitoring the success and failures of the tasks.

2.3 Why MapReduce on VMs

Currently, driven by the increasing maturity of virtualization technology in general, virtual machine in particular, VMs have been experienced in various distributed systems such as grid [11], HPC application [12-14]. To this end, in this section we

discuss the main factors contributing to the interests of MapReduce on virtual machines:

1. Driven by the increasing popularity of cloud computing in which VMs are the main computation units, and the widely adoption of MapReduce, due to its magnificent features, as the programming model for data intensive applications. Consequently, combining theses two technologies is promising approach for large scale data cloud computing. Moreover, VMs can be effectively used to utilize the cluster resources; especially those equipped with multi-core processors and can greatly benefit cluster computing from aspects of ease of management, customized OS and security [11].
2. Recently, MapReduce is using speculative tasks approach to provide reliable performance. Speculative tasks are normally performed by re-execute the task on different DataNodes. Executing two copies of the same tasks can cause waste of the cluster resources. Thus, benefiting of the recent advances of VM checkpointing and live migration [17, 18], it is feasible to use these techniques to improve the reliability of MapReduce performance. Moreover, VM checkpointing and migration can be used to improve the reliability of the MapReduce master node as it is single point of failure.

3 Methodology and Hardware Platform

Our experimental hardware consists of seven nodes cluster. Each node in the cluster is equipped with two quad-core 2.33GHz Xeon processors, 8GB of memory and 1TB of disk, runs RHEL5 with kernel 2.6.22, and is connected with 1 Gigabit Ethernet. In VM-based environments, we use Xen 3.2 [19]. The VMs are running with RHEL5 with kernel 2.6.22. VM is configured with 1 VCPU and 1GB memory. The same cluster is used to obtain performance results for both the VM-based environment and the native, non-virtualized environment. All results described in this paper are obtained using Hadoop version 0.18.0, while the data is stored with 2 replicas per block in HDFS. We perform five jobs per experiment and compute the average across jobs.

3.1 Experiments Design and Motivations

This section reports on several experiments designed to evaluate the MapReduce performance on virtual machines. First, as the HDFS playing a big role during the MapReduce process, we comparatively evaluate the performance of the HDFS when writing/reading data in both physical and virtual cluster. Second, we report on the feasibility of using VM to enhance the performance of MapReduce by increasing the resource utilization as CPU cycles. Third, we conduct the experiments to evaluate the execution time and the number of the lunched speculative tasks with different VMs load per physical machine.

3.2 Benchmarks

In all our experiments, we use two different and widely used benchmarks, sort and wordcount benchmarks, which are sample programs in the Hadoop distribution.

- Sort Benchmark. The sort benchmark [20] simply uses the map/reduce framework to sort the input directory into output directory (two replica by default), both the input and output must be sequence files. The map function extracts the key from each record and emits a <key, record> pair, the reduce function emits all pairs unchanged. All the input data are generated using the Random Writer sample application in the Hadoop distribution.
- WordCount Benchmark. The wordcount [20] counts the number of occurrences of each word in a file and writes the output to local disk. The map function emits each word plus an associated count of occurrences. The reduce function sums together all counts emitted for a particular word. In addition, a combiner function is used to fold redundant <word, _> pairs into a single one, which magnificently reduces the network I/O. All the input data are generated by duplicating the text file used by Phoenix [5].

4 Experiment Results

In this section, we evaluate the performance impact of using virtualization for the two specific benchmarks on our cluster system.

4.1 Hadoop Distributed File System

MapReduce programming model strongly depends on the underlying storage system, namely GFS for Google MapReduce and HDFS in Hadoop. We evaluate the HDFS performance in both physical cluster (PH-HDFS for short) and virtual cluster (VM-HDFS for short) when transferring data to and from the DFS, using the *put* and *get* command respectively. In particular, we conduct our experiments using three different scenarios (different data size, different cluster size, and different throughput when multi requests). In all our experiments one VM has been uniquely deployed on each physical node.

First, we evaluate the performance when transferring different data size (1.5GB, 3GB, 6GB, and 12 GB). As shown in Fig. 1, the PH-HDFS performs better than VM-HDFS in terms of reading and writing capacities. Moreover, the performance gap is markedly increases as the data size is increasing in both cases writing data to or reading data from the DFS.

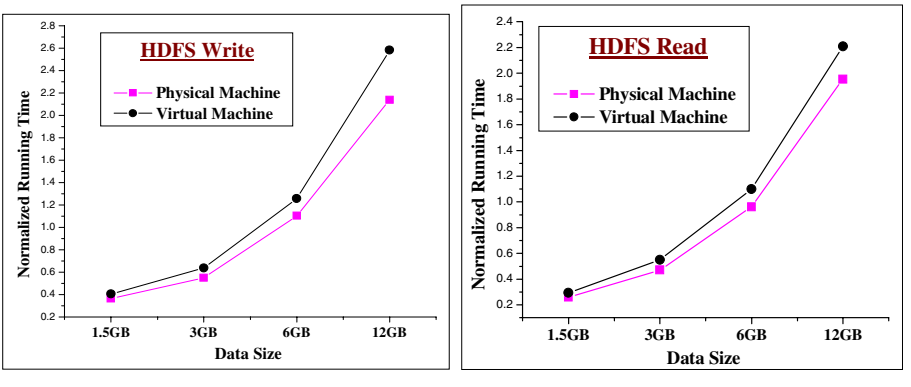

Fig. 1. PH-HDFS *vs* VM-HDFS with different data scale and 7 nodes cluster

Second, we fix the data distribution per node to 512MB. Accordingly, if there are 2, 4 and 6 DataNodes, 1, 2 and 3GB of data are transferring respectively. The PH-HDFS also performs better than the VM-HDFS as the number of data node increases as shown in Fig. 2.

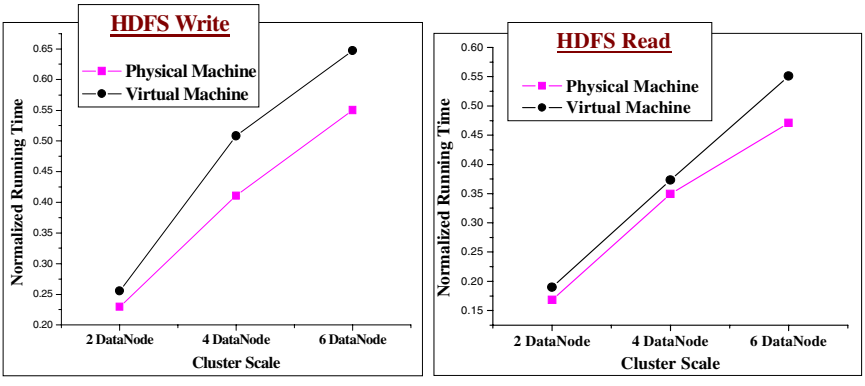

Fig. 2. PH-HDFS *vs* VM-HDFS with different cluster scale, with the same data distribution (512MB per DataNode)

Third, we evaluate different throughputs by starting 1, 2, and 3 requests simultaneously and measuring the time needed for data transfer, the average in the case of two and three requests. As shown in Fig. 3, the PH-HDFS performs better than VH-HDFS. In addition the performance gap is markedly increasing when writing data in, while it is slightly increasing in case of reading data from the DFS.

4.2 VMs Feasibility

Driven by the advent of multi-core, it is feasible to study the opportunities of utilizing the multi-core processor and memory management of the cluster while processing large scale data using Hadoop.

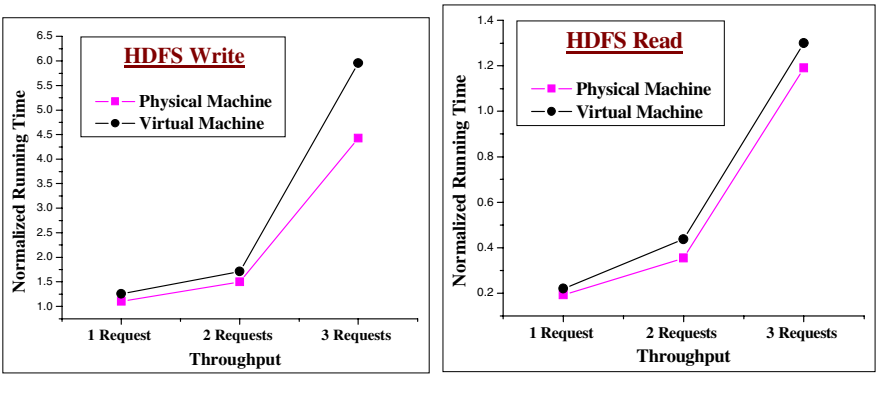

Fig. 3. PH-HDFS *vs* VM-HDFS with different throughput (requests/s) and 7 nodes cluster

In our experiment, we choose the wordcount benchmark because the data transfer during the copy phase is small and this will reduce the effects of data transfer in our experimental results. We evaluate the performance among four homogenous clusters as shown in Table 1, physical cluster (Ph-Cluster) and three virtual clusters (V-Cluster, V2-Cluster, and V4-Cluster with one, two, and four VMs running on each physical machine, respectively).

Table 1. Four Homogeneous Testbed Clusters

	Ph-Cluster	V-Cluster	V2-Cluster	V4-Cluster
VM Load	-	1 VM/Node	2 VM/Node	4 VM/Node
Cluster Size	7 Physical nodes	7 VM nodes	13 VM nodes	25 VM nodes

As shown in Fig. 4, the wordcount job in Ph-Cluster costs less time than in V-Cluster. In particular, when computing 1GB, the performance gap is small. While for data set of 8GB, this gap is obviously big. This is because: (1) HDFS performing better in physical cluster than in virtual one as shown in section 4.1, and (2) the increasing number of speculative tasks causing inefficient utilize of the resources. On the other hand, as expected, V2-Cluster and V4-Cluster are performing faster than the Ph-Cluster. This is because more computing cycles are available and more slots are free.

4.3 MapReduce on Virtual Machines Performance Analyze

In this section we report on different experiments with different VM load per physical node as shown in Table 1, and different data distribution on each data node.

As shown in Fig. 5, when running the sort benchmark, the execution time of the jobs for the same data distribution increases with the increment of VMs deployed on each physical node. Moreover, the performance gaps among these three different

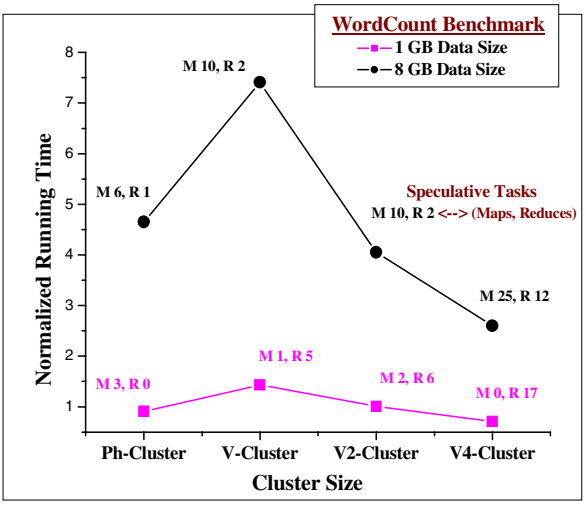

Fig. 4. Wordcount execution: physical cluster *vs* virtual clusters with different VM load per node (1, 2, and 4), and two data sets (1GB and 8 GB)

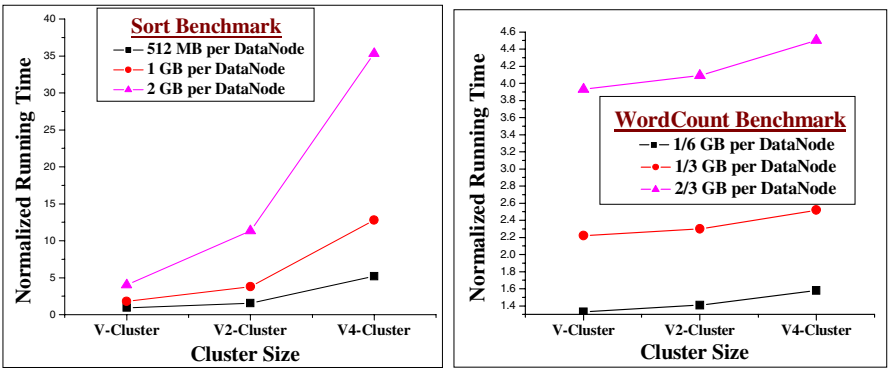

Fig. 5. Sort and wordcount execution with different VM load per node, and different data size

clusters markedly increase as the size of data distribution increases. This is due to three reasons: (a) the bad performance of HDFS on VMs when reading/writing the data blocks from/to the HDFS; (2) the increasing number of speculative tasks as shown in Fig. 6, and (3) more importantly, the large amount of data transferred during the copy of the intermediate data, especially that VMs are competing for the node I/O resources.

For the wordcount benchmark this gap is slightly increasing for different data distribution, caused by the same aforementioned reasons with less emphasize on the third reason as the data transfer is small.

5 Discussion and Open Issues

Based on the experience and the above experiments with MapReduce on VMs, we draw several open issues:

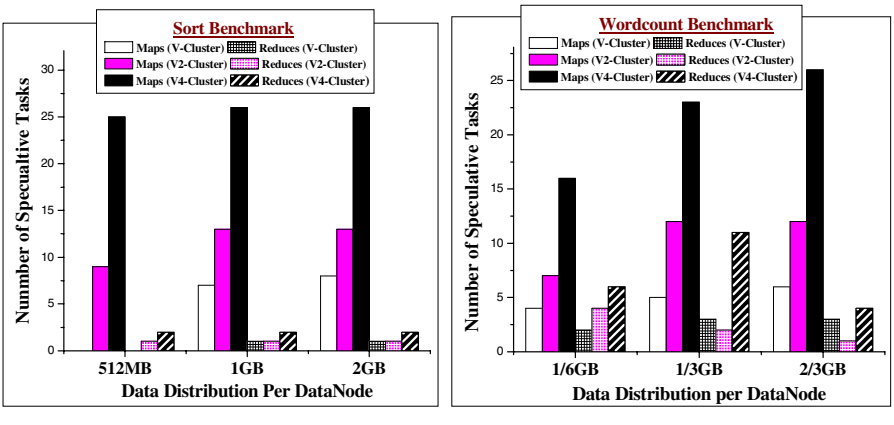

Fig. 6. Sort and wordcount launched speculative tasks with different VM load per node, and different data distribution

- Motivated by our experiments results in section 4.1, as well as VM being highly prone to error, it is useful to separate the permanent data storage (DFS) from the virtual storage associated with VM. In addition, using VM as execution unit only will allow us to study the possibilities of using VM migration as a replacement of the existing fault tolerance mechanism represented as speculative tasks, resulting with better performance and reduce the wasted resources caused by the increased number of speculative tasks in VM-based cluster [21] as shown in section 4.3.
- As shown is section 4.2, it is feasible to use VM in data intensive computing systems to fully utilize the physical node resources, using VM only as a computation unit for the data located on its physical node. More importantly, VM should be configured to perfectly suit the running data intensive applications.
- VMs within the same physical node are competing for the node I/O, causing poor performance. Therefore, many possibilities could be studied to improve it such as starting reduce tasks after all the map tasks are successfully finished and make the data transfer physical machine-based by collecting all the maps output within one physical node, from different VMs, and check new scheduling algorithms to reduce the data transferred.
- Finally, as the master node is a single point of failure for the Map/Reduce infrastructure, if it goes down, all running jobs are lost. Therefore, VM being highly prone to failure, it is not highly recommended to keep the master node physical based or use the VM checkpointing to implement more reliable master.

6 Conclusion and Future Work

Driven by the new trend in distributed system towards cloud computing and the increasing popularity and adoption of cloud storage services, processing this large data has become big challenge. Data-aware based data intensive approach is proven to be efficient and robust, where data and computation are collocated. This approach has been widely used and studied, especially after the huge success of Google version, namely GFS and MapReduce. Meanwhile, through the recent improvement and maturity of virtual machine, cloud community is strongly adopting VMs as the computation units in the cloud. Lots of research have been carried out about VM-based HPC application, while up to date, no paper has introduced and evaluated the integration of VM and MapReduce based data intensive application, mainly the challenges and the opportunities.

Based on our experiments with MapReduce, Hadoop in particular, we have elaborated a number of issues when running data intensive applications using Hadoop in virtual cluster. This is intended as an invitation to cloud researchers to influence these important technologies in a constructive manner by drawing on research and experience.

Current and future research efforts include developing MapReduce framework on VMs, namely Cloudlet [22], and experimenting with scheduling and migrating the VMs within a cluster to improve high performance, reliability, manageability, and power management.

References

1. Szalay, A., Bunn, A., Gray, J., Foster, I., Raicu, I.: The Importance of Data Locality in Distributed Computing Applications. In: Proceedings of the NSF Workflow Workshop (2006)
2. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. In: Proceedings of 19th ACM Symposium on Operating Systems Principles, pp. 29–43. ACM Press, New York (2003)
3. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: Proceedings of 6th Conference on Operating Systems Design & Implementation (2004)
4. Hadoop, http://lucene.apache.org/hadoop
5. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating MapReduce for Multi-core and Multiprocessor Systems. In: Proceedings of 13th International Symposium on High Performance Computer Architecture, pp. 13–24. ACM Press, New York (2007)
6. Bryant, R.E.: Data-Intensive Supercomputing: The Case for DISC. CMU-CS-07-128, Technical Report, Department of Computer Science, Carnegie Mellon University (May 2007)
7. Chen, S., Schlosser, S.W.: Map-Reduce Meets Wider Varieties of Applications, IRP-TR-08-05, Technical Report, Intel. Research Pittsburgh (May 2008)
8. CNET news, http://news.cnet.com/8301-13505_3-10196871-16.html (accessed September 2009)
9. Amazon Elastic Cloud Computing, http://aws.amazon.com/ec2/
10. GoGrid Cloud Hosting, http://www.gogrid.com/
11. Figueiredo, R., Dinda, P., Fortes, J.: A Case for Grid Computing on Virtual Machines. In: Proceedings of 23rd International Conference on Distributed Computing Systems, pp. 550–559. IEEE CS Press, Los Alamitos (2003)
12. Mergen, M.F., Uhlig, V., Krieger, O., Xenidis, J.: Virtualization for High Performance Computing. ACM SIGOPS Oper. Syst. Rev. 40(2), 8–11 (2006)
13. Huang, W., Liu, J., Abali, B., Panda, D.K.: A Case for High Performance Computing with Virtual Machines. In: Proceedings of 20th ACM International Conference on Supercomputing, pp. 125–134. ACM Press, New York (2006)
14. Nagarajan, A.B., Mueller, F., Engelmann, C., Scott, S.L.: Proactive Fault Tolerance for HPC with Xen Virtualization. In: Proceedings of 21st ACM International Conference on Supercomputing, pp. 23–32. ACM Press, New York (2007)
15. Amazon Elastic MapReduce, http://aws.amazon.com/elasticmapreduce/
16. Amazon Simple Storage Service, http://aws.amazon.com/s3/
17. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live Migration of Virtual Machines. In: Proceedings of USENIX Symposium on Networked Systems Design and Implementation (2005)
18. Zhao, M., Figueiredo, R.J.: Experimental Study of Virtual Machine Migration in Support of Reservation of Cluster Resources. In: Proceedings of 2nd International Workshop on Virtualization Technology in Distributed Computing (2007)
19. XenSource (2008), http://www.xensource.com/
20. Hadoop Wiki (2008), http://wiki.apache.org/hadoop/
21. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving mapreduce performance in heterogeneous environments. In: Proceedings of 8th USENIX Symposium on Operating Systems Design and Implementation (2008)
22. Ibrahim, S., Jin, H., Cheng, B., Cao, H., Wu, S., Qi, L.: Cloudlet: Towards MapReduce implementation on Virtual machines. In: Proceedings of 18th ACM International Symposium on High Performance Distributed Computing, pp. 65–66. ACM Press, New York (2009)