Performance Analysis of Hadoop MapReduce on an OpenNebula Cloud with KVM and OpenVZ Virtualizations

Pedro Roger Magalhães Vasconcelos
Federal University of Ceará
Ceará, Brazil
Email: pedro.roger@alu.ufc.br

Gisele Azevedo de Araújo Freitas
Federal University of Ceará
Ceará, Brazil
Email: gisele@lia.ufc.br

***Abstract*—Cloud computing provides access to a set of resources such as virtual machines, storage and network as services. In this context, virtualization has been used as an platform for resource-intensive applications, like Hadoop, as it has brought features like server consolidation, scalability and better resources usage. OpenVZ and KVM are very popular and widely used virtualization platforms with distinct virtualization modes: containet-based and full-virtualization. In this work, we conducted benchmarks to measure the performance of a Hadoop cluster deployed on OpenNebula clouds with KVM and OpenVZ. Our results show that OpenVZ performs better than KVM in the CPU and I/O reading benchmarks. KVM achieves better performance in the I/O writing benchmarks.**

***Keywords*—Big Data, MapReduce; cloud computing; virtualization; KVM; OpenVZ.**

I. Introduction

Cloud Computing is creating an environment suitable for web services. Nowadays, the most common architecture is 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, where data centers are owned and maintained by third party. However, in terms of resources, the three main characteristics of cloud are: on-demand unlimited data storage, on-demand computation power with no lock, mainly represented as Virtual Machines (VMs), and using internet, limited bandwidth connection, to access, use and process these resources.

Typically a cloud is hosted on a server farm with a large amount of clustered computers. These provide the hardware resources. The cloud provider (the organization that hosts the servers) offers an interface for users to pay for a certain amount of processing power, storage or computers in a business model. These resources can then be increased or decreased based on demand, so the user only needs to focus on it's contents whereas the provider takes care of maintenance, security and networking [1].

The servers in the server farm are usually virtualized, although they are not required to be to actually be included in a cloud. Virtualization is a ground pillar in cloud computing; it enables the provider to maximize the processing power on the raw hardware and gives the cloud elasticity, the ability for users to scale the instances required. It also helps providing two other key features of a cloud: multitenacity, the sharing of resources, and massive scalability, the ability to have huge amounts of processing systems and storage areas (tens of thousands of systems with large amounts of terabytes or petabytes of data) [2].

Virtualization is a logical representation of the computer using a software [3]. In a physical computer, there is a single operating system running one or more applications. Using virtualization, a software which runs on a single physical computer consist by physical resources running into different virtual machines. Hence several operating systems can be run inside a single machine. A result from a virtual computing do not affect other virtual computing. The main advantage of virtualization is the effective use of hardware. That is, sharing resources to each virtual machine, there has been complete utilization of the hardware. Billions of dollars have been invested on the research on controlling heat dissipation in data center. The only way is to use the less number of servers, hence virtualization on server allows less physical hardware and less dissipation of heat. Nowadays, there has been a significant increase in the cost of the hardware, hence virtualization allows fewer physical hardware and hence reduced cost. Some of the parameters which adds up to the cost saving are easier maintenance and lesser electricity. Redeployment and backups are made easier in virtualization by using a snapshot mechanism [4].

Nowadays, there are lot of an open source cloud computing solutions for providing infrastructure environments that include public, private or hybrid clouds such as Eucalyptus [5], OpenNebula [6], and CloudStack [7]. OpenNebula is an open source solution that allows easily deploy private/hybrid infrastructure clouds based on IaaS model. All the physical hosts on the cloud do not need to have homogeneous configuration, but it is possible to use different hypervisors on different GNU/Linux distributions on a single OpenNebula cluster.

OpenNebula has a great flexibility regarding hypervisor usage since it natively supports KVM/Xen (which are open source) and VMware ESXi. Drivers provided by the OpenNebula community provides support for OpenVZ operating system-level virtualization.

The Kernel-based Virtual Machine (KVM) [8] is a full-virtualization solution for the Linux kernel. KVM requires a processor with hardware virtualization extension. In full virtualization, a layer, commonly called the hypervisor or

978-1-908320-39/1/$25.00©2014 IEEE

the virtual machine monitor, exists between the virtualized operating systems and the hardware. This layer multiplexes the system resources between competing operating system instances.

A hypervisor is composed of several components usually having to write a scheduler, memory management, I/O-stack for a new hypervisor, as well as drivers for the architecture on which the hypervisor is targeted at. KVM unlike other hypervisors such as Xen, has focused the implementation on the guest handling of the virtualization, letting the Linux kernel operate as the hypervisor. Since Linux has developed into a secure and stable operating system, as well as having some of the most important features for a hypervisor, such as a scheduler and a plethora of drivers, it is more efficient to reuse and build upon this rather than reinvent a hypervisor [9].

OpenVZ (Open Virtuozzo) [10] is an operating system-level virtualization technology based on the Linux kernel and operating system. OpenVZ allows a physical server to run multiple isolated operating system instances, known as containers, Virtual Private Servers (VPSs), or Virtual Environments (VEs).

Operating system-level virtualization is type of server virtualization technology which works at the OS layer. The physical server and single instance of the operating system is virtualized into multiple isolated partitions, where each partition replicates a real server. The OS kernel will run a single operating system and provide that operating system functionality to each of the partitions.

While hypervisor-based virtualization provides abstraction for full guest OS's (one per virtual machine), operating system-level virtualization works at the operation system level, providing abstractions directly for the guest processes. In practice, hypervisors work at the hardware abstraction level and operating system-level virtualization at the system call layer [11].

This paper evaluates the performance of a Hadoop MapReduce cluster on a OpenNebula cloud under two different types of virtualization: full virtualization and operating system-level virtualization. Results of various Hadoop benchmarks under these virtualization types are presented here.

The rest of the paper is organized as follows. Section 2 discusses the MapReduce model and the Hadoop framework. Section 3 discusses the related works. Section 4 presents the experiment setup and the benchmarking tools. Section 5 presents and discusses the results of the experiment. Finally, Section 6 concludes the study.

II. MapReduce

MapReduce (MR) [12] is a programming model that works on large datasets. Many organizations use MapReduce model for computing when they have huge datasets and need to process them within short time. In Facebook, event logs from its website are imported into a 600-node Hadoop datawarehouse, where they are used for a variety of applications, including business intelligence, spam detection, and ad optimization. The warehouse stores 2 PB of data, and grows by 15 TB per day. In addition to production jobs that run periodically, the cluster is used for many experimental jobs, ranging from multi-hour machine learning computations to 1-2 minute ad-hoc queries submitted through an SQL interface to Hadoop called Hive [13]. The system runs 7500 MapReduce jobs per day and is used by 200 analysts and engineers [14].

MapReduce works by breaking the processing into two phases: the map phase and the reduce phase [15]. In MapReduce, the Map function processes the input in the form of key/value pairs and generate intermediate key/value pairs, and the Reduce function process all intermediate values associated with the same intermediate key generated by the Map function [15].

MapReduce [16] performs a series of operations on the input key and value. As many companies now adopt Hadoop, the requirements of what input varies from flat files to databases. In case of database, one can take an example of rows from the table.

Map functions are distributed between all nodes in cluster and locality of the data can be considered unknown. There are also better ways to balance the data and jobs that are local to the data.

Job scheduling in Hadoop is performed by a master, which manages a number of slaves. Each slave has a fixed number of map slots and reduce slots in which it can run tasks. Typically, administrators set the number of slots to one or two per core. The master assigns tasks in response to heartbeats sent by slaves every few seconds, which report the number of free map and reduce slots on the slave. Hadoop's default scheduler runs jobs in FIFO order, with five priority levels. When the scheduler receives a heartbeat indicating that a map or reduce slot is free, it scans through jobs in order of priority and submit time to find one with a task of the required type [14].

A. Hadoop MapReduce and Hadoop Distributed File System

Hadoop MapReduce is a distributed programming framework and an execution environment for MapReduce programs. The execution environment also includes a job scheduling system that coordinates the execution of multiple MapReduce programs, which are submitted as batch jobs. A MapReduce job consists of multiple map and reduce tasks that are scheduled to run in the Hadoop cluster nodes. Multiple jobs can run simultaneously in the same cluster. There are two types of nodes that control the job execution process: a JobTracker and a number of TaskTrackers [11].

The Hadoop Distributed File System (HDFS) [16] is the mains storage system used by Hadoop. HDFS creates multiple replicas of data blocks and distributes them among the cluster nodes. It is a distributed file system with similarities to other file systems, but with some differences. The HDFS is highly fault tolerant and is designed to run on low-cost hardware. Provides high performance processing large volumes of data. From end user perspective, the HDFS is seen as a traditional file system.

All application data in Hadoop is stored as HDFS files, which are composed of data blocks of a fixed size (64 MB each, by default) distributed across multiple nodes. There are two types of nodes in a HDFS cluster: a NameNode and a number of DataNodes. The NameNode maintains the file system metadata, which includes information about the files

and directories tree as well as where each data block is physically stored. DataNodes store the data blocks themselves. Every time a client needs to read a file from HDFS, it first contacts the NameNode to determine the DataNodes where all the blocks for that file are located. Then, the client starts reading the data blocks directly from the DataNodes [11].

The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later [17].

The placement of replicas is critical to HDFS reliability and performance. Optimizing replica placement distinguishes HDFS from most other distributed file systems. This is a feature that needs lots of tuning and experience. The purpose of a rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization.

Large HDFS instances run on a cluster of computers that commonly spread across many racks. Communication between two nodes in different racks has to go through switches. In most cases, network bandwidth between machines in the same rack is greater than network bandwidth between machines in different racks.

To minimize global bandwidth consumption and read latency, HDFS tries to satisfy a read request from a replica that is closest to the reader. If there exists a replica on the same rack as the reader node, then that replica is preferred to satisfy the read request. If a HDFS cluster spans multiple data centers, then a replica that is resident in the local data center is preferred over any remote replica [17].

III. Related Work

Virtualization is getting more and more popular in a cloud environment, an example is the Amazon elastic MapReduce [18], [19]. In paper [19], the authors propose a cloud based MapReduce, which is offered by the Amazon web services called as elastic MapReduce (EMR). This EMR allows user to sign up to the Amazon web service and after getting sign in, user can submit their MapReduce jobs using EMR API which is developed by some programming models like python or java. These MapReduce jobs are then sent to the Hadoop cluster, which consist of three virtual machines. That is, Unique master VM, which acts as HDFS and schedules MapReduce task over other VMs, Multicore VMs which produces storage for HDFS and computes all the MapReduce tasks. Finally a multitask VMs which don't store any data but executes MapReduce task.

In [20], Chen He, Derek Weitzel, David Swanson and Ying Lu discussed experiments by running Hadoop MapReduce on a grid which provides a free, elastic, and dynamic MapReduce environment on the opportunistic resources of the grid. For Hadoop evaluation, they successfully extended Hadoop to 1100 nodes on Grid. From the evaluation, Chen and his team found that the unreliability of the grid makes Hadoop on the grid very challenging. K-means Clustering Using Hadoop Map Reduce by Grace Nila Ramamoorthy [21] evaluated the performance of K-Means clustering using Hadoop Map Reduce on standalone servers infrastructure. This model works well, but when it comes to scalability and efficient use of resources and need for rebuilding environment for different projects it becomes complex and very time consuming.

IV. Experimental evaluation

To evaluate the performance of Hadoop cluster on each virtualization platform, we propose the establishment of two private OpenNebula clouds. The execution environments consists of two identical servers IBM BladeCenter HS21 with two Intel Xeon CPUs E5-2620 of 2.00GHz (with 6 cores and HT technology in each), 48GB of RAM, connected to a SAN via fiber-channel, and interconnected through a GigabitEthernet network adapter and switch. The hosts systems were installed on an Ubuntu GNU/Linux 14.04.1 LTS amd64 system. The OpenNebula version used was 4.8.0.

Each Hadoop cluster consists of six virtual machines, one master and five slaves, with the same configuration: Ubuntu GNU / Linux amd64 LTS 12.04, 2 vCPUs, 2GB of vRAM, 1GB swap and 10GB of disk.

The version of QEMU/KVM is 2.0.0. For best performance, virtual machines used KVM VirtIO paravirtualized drivers for disk and network. For OpenVZ, the kernel used was 2.6.32-openvz-amd64-042stab093.5.

We used Hadoop version 1.2.1 and Oracle Java version 1.7.0_45 in all virtual machines. The HDFS file system had 60GB (6 x 10GB per virtual machine) of distributed storage, HDFS block size was 64MB and replication factor was set to 3. The task timeout was set to 30 minutes. The following sections describe the benchmarks.

A. Wordcount

The WordCount program is an application that reads text files as input and computes the number of occorences of each word in a file. The output file is a text file where each line contains a word and the number of times it occurred, separated by a tab.

Each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word and 1. Each reducer sums the counts for each word and emits a single key/value with the word and sum. As an optimization, the reducer is also used as a combiner on the map outputs. This reduces the amount of data sent across the network by combining each word into a single record [22].

The WordCount already comes with the Hadoop default installation and is widely used as a method of comparing performance between different Hadoop clusters.

The input files used in the benchmark were generated from the concatenation of random text files downloaded from Project Gutenberg [23] to the desired sizes: 64MB, 128MB, 256MB, 512MB, 1GB and 2GB.

As can be observed in Fig. 1, OpenVZ reached the lowest execution time of WordCount.

B. TeraSort

The goal of TeraSort benchmark is to sort certain volume of data as quickly as possible. It is a benchmark that combines the use of HDFS layer and MapReduce layer.

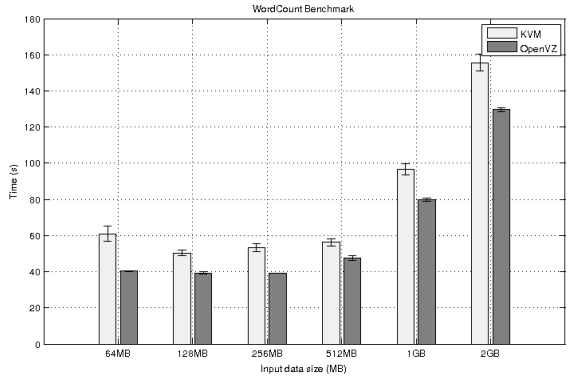

Figure 1: Wordcount Results

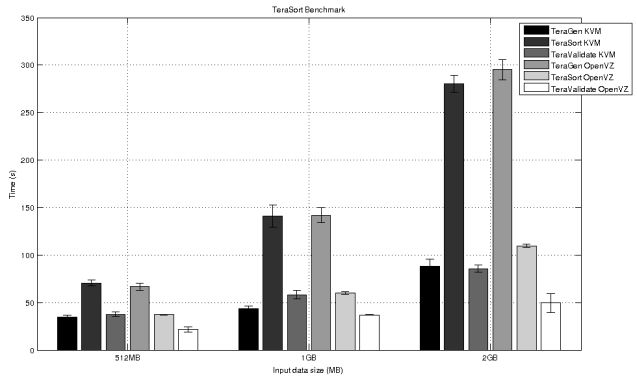

Figure 2: TeraSort Results

TeraSort consists of 3 map/reduce applications:

- **TeraGen** is a map/reduce program to generate the data.
- **TeraSort** samples the input data and uses map/reduce to sort the data into a total order.
- **TeraValidate** is a map/reduce program that validates the output is sorted.

TeraGen generates output data that is byte for byte equivalent to the C version including the newlines and specific keys. It divides the desired number of rows by the desired number of tasks and assigns ranges of rows to each map. The map jumps the random number generator to the correct value for the first row and generates the following row [24].

TeraSort is a standard map/reduce sort, except for a custom partitioner that uses a sorted list of N-1 sampled keys that define the key range for each reduce. In particular, all keys such that sample[i-1] $\leq$ key $<$ sample[i] are sent to reduce i. This guarantees that the output of reduce i are all less than the output of reduce i+1. To speed up the partitioning, the partitioner builds a two level trie that quickly indexes into the list of sample keys based on the first two bytes of the key. TeraSort generates the sample keys by sampling the input before the job is submitted and writing the list of keys into HDFS. The input and output format, which are used by all 3 applications, read and write the text files in the right format. The output of the reduce has replication set to 1, instead of the default 3, because the contest does not require the output data be replicated on to multiple nodes [24].

TeraValidate ensures that the output is globally sorted. It creates one map per a file in the output directory and each map ensures that each key is less than or equal to the previous one. The map also generates records with the first and last keys of the file and the reduce ensures that the first key of file i is greater that the last key of file i-1. Any problems are reported as output of the reduce with the keys that are out of order [24].

We used variable sizes for generation of input data through Teragen: 512MB, 1GB and 2GB.

The Figure 2 shows the performance of the two clouds in the execution of a variable size TeraSort run.

C. TestDFSIO

The TestDFSIO benchmark is a read/write test to HDFS. It is useful to perform stress testing in the HDFS, to test parameters in NameNodes and DataNodes, to find bottlenecks and evaluate the cluster I/O rate. The TestDFSIO consists of two tests: the first, generates and writes the files in a HDFS directory; the second reads the files created by the first run and performs measurements.

Each file is read or written in a separate map task, and the output of the map is used for collecting statistics relating to the file just processed. The statistics are accumulated in the reduce to produce a summary.

In our runs, we inform the TestDFSIO to create 10 files of 100MB in HDFS. Fig. 3 demonstrates the performance of the two platforms during the benchmark. As we see, OpenVZ performance in writing tests was much lower than KVM. Although, in the reading tests OpenVZ performs better than KVM.

D. NNBench

The NNBench is useful for testing the hardware and configuring the NameNode. This test generates various requests to the HDFS with normally very small payloads for the purpose of stressing it. The benchmark can simulate requests for creating, reading, renaming and deleting files on HDFS. In our implementation, NNBench created 1000 files using 6 maps and 2 reducers. Figure 4 describes the NNBench runtimes for each cloud.

E. MRBench

The MRBench loops a small job a number of times. As such it is a very complimentary benchmark to the large-scale TeraSort benchmark suite because MRBench checks whether small job runs are responsive and runnig efficiently in the cluster. It put its focus on the MapReduce layer as its impact on the HDFS layer is very limited.

We ran MRBench in order to run a loop of 50 small test jobs. Fig. 5 shows the elapsed time for each one of the clouds.

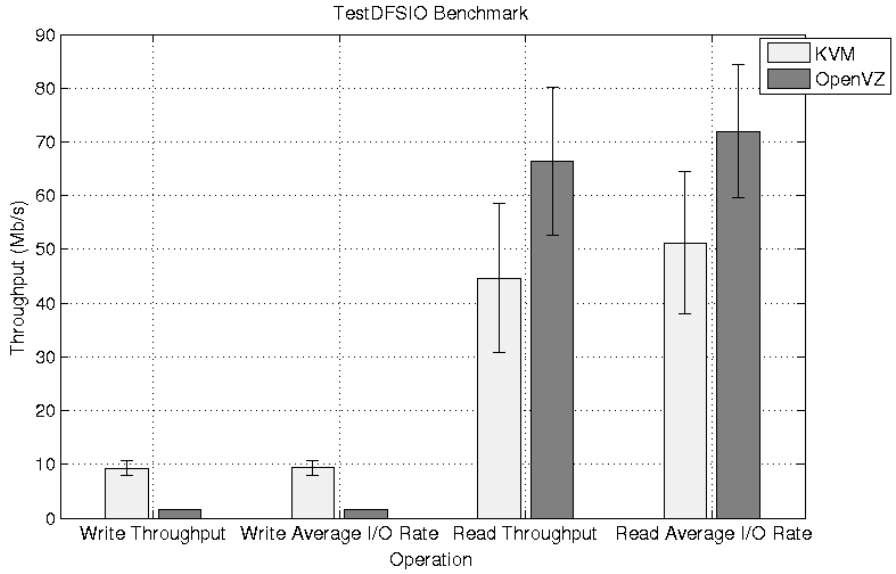

Figure 3: TestDFSIO Results

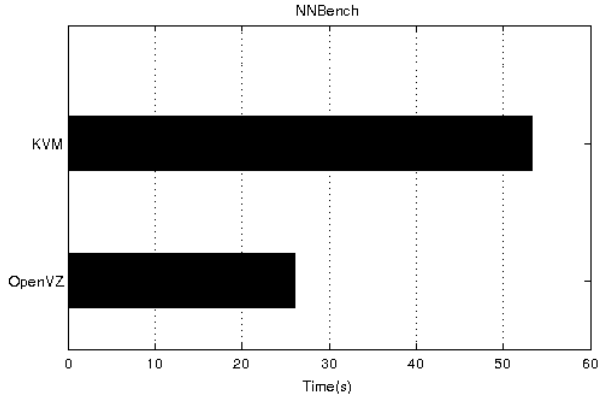

Figure 4: NNBench Results

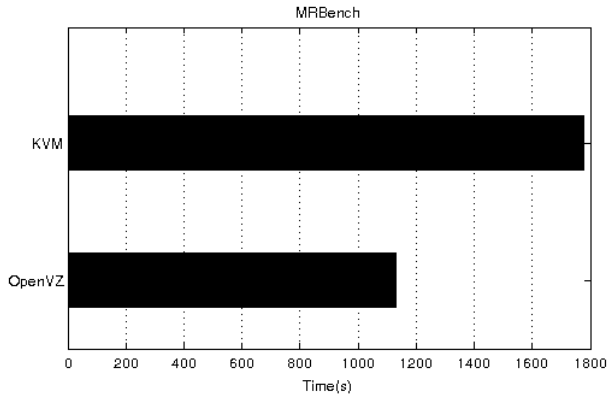

Figure 5: MRBench Results

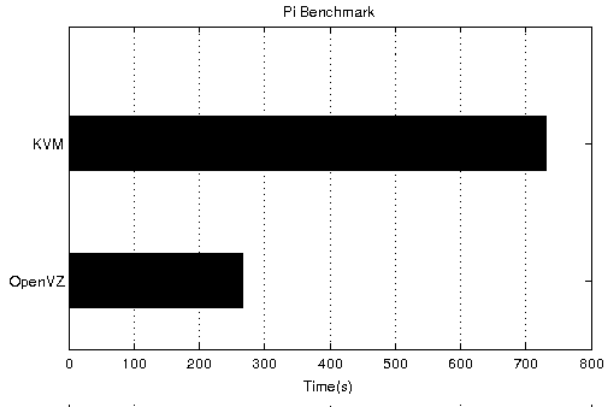

Figure 6: Pi Benchmark Results

F. Pi

The Hadoop Pi benchmark is a map/reduce program that estimates Pi using monte-carlo method. This benchmark is focused on computation and involve nearly no storage I/O or network traffic. Fig. 6 shows the Pi execution times for runs that calculates 10 billions samples spread across 6 maps tasks.

V. Discussion

From the benchmarks results we can observe that OpenVZ performs better than KVM on CPU intensive tests like Word-Count, TeraSort, TeraValidate, MRBench and Pi. Openvz too reachs better results than KVM on I/O reading tests as showed in the values of Read Throughput and Read Average I/O Rate operations of DFSIOtest benchmark. However, OpenVZ performs worst than KVM in I/O writing tests of large files in TeraGen and reached low rates of Write Throughput and Write Average I/O in TestDFSIO. But, in the sequential creation of

inumerous small files in NNBench test the time elapsed in KVM run was almost twice of OpenVZ time.

VI. Conclusion

In cloud computing environments, virtualization had an important contribution, by allowing server consolidation and ease of deploy and management. The ability of cloud computing environments in supporting workloads have grown in the last years due of the advances in hypervisors and hardware's improves like virtualization-specific instructions sets.

A cloud can be shared among many users or institutes, which may have different requirements in terms of software packages and configurations. As presented, such platforms might benefit from using virtualization technologies to provide better resource sharing and custom environments.

KVM and OpenVZ are widely used solutions for virtualization under distinct approaches, and a resource-intensive application, like Hadoop needs that the performance overhead be reduced to be able to take advantage of virtualization systems.

According to our results, OpenVZ reach better rates and execution times than KVM on CPU intensive tests, I/O reading and small files I/O writing tests. KVM performed better results than OpenVZ only on the I/O writing tests of large files.

By using OpenVZ, an Hadoop cluster can achieve a high performance on virtualized systems when running jobs that use intensively CPU, network and I/O reading. Jobs that perform intense disk writing operations should be executed with caution or executed natively.

The overall effectiveness of OpenVZ allied to cloud computing features of OpenNebula brings OpenNebula OpenVZ suite as an effective Cloud Computing platform for Hadop MapReduce clusters.

Acknowledgment

The authors would like to thank the Information Technology Center of the State University Vale do Acaraú (UVA) for allowing the use of their computing infrastructure for the deployment, development and testing of the presented work.

The authors are thankful too to FUNCAP (Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico) for the support they have received to make the presentation of this work.

References

[1] J. Nilsson and P. Johansson, *Hadoop MapReduce in Eucalyptus Private Cloud.*, Bachelor's Thesis in Computing Science. Umea, Sweden, 2011.

[2] T. Mather and S. Kumaraswamy and S.Latif, *Cloud Security and Privacy*. O'Reilly, 2009.

[3] IBM, *Virtualization in education*, October 2007.

[4] A. Devadiga, P.R Shalini and A. K. Sinha, *Virtual Hadoop: The Study and Implementation of Hadoop in Virtual Environment using CloudStack KVM*, International Journal of Engineering Development and Research (IJEDR), volume 2, issue 2, 2014.

[5] Eucalyptus, *Eucalyptus — Open Source Private Cloud Software*, http://www.eucalyptus.com/, (Access Date: 04 Dec., 2014).

[6] OpenNebula.org, *OpenNebula — Flexible Enterprise Cloud Made Simple*, http://opennebula.org/, (Access Date: 04 Dec., 2014).

[7] CloudStack, *Apache CloudStack: Open Source Cloud Computing*, http://cloudstack.apache.org/, (Access Date: 04 Dec., 2014).

[8] KVM, *Main Page - KVM*, http://www.linux-kvm.org/page/Main_Page, (Access Date: 04 Dec., 2014).

[9] J. Magnus and G. Opsahl, *Open-source virtualization. Functionality and performance of Qemu/KVM, Xen, Libvirt and Virtualbox.*, Master's Thesis. Oslo, Norway, 2013.

[10] OpenVZ, *OpenVZ Linux Containers*, http://openvz.org/Main_Page, (Access Date: 04 Dec., 2014).

[11] Miguel G. Xavier and Marcelo V. Neves and Cesar A. F. De Rose,*A Performance Comparison of Container-based Virtualization Systems for MapReduce Clusters*, 22nd EuroMicro International Conference On Parallel, Distributed and network-based Processing (PDP2014), TuRin, Italy, IEEE, Feb 2014.

[12] Apache Hadoop, *MapReduce tutorial Apache Hadoop 1.2.1 documentation by Hadoop wiki*, http://hadoop.apache.org/docs/r1.2.1/, (Access Date: 04 Dec., 2014).

[13] Apache, *Apache Hive TM*, https://hive.apache.org/, (Access Date: 04 Dec., 2014).

[14] M. Zaharia and D. Borthakur and J. S. Sarma and K. Elmeleegy and S. Shenker and I. Stoica,*Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling*, 5th European conference on Computer systems (EuroSys '10), New York, USA, 265-278.

[15] T. White, *Hadoop - The Definitive Guide*, O'Reilly Media/Yahoo Press, 2nd edition, 2010.

[16] J. P. Jacob and A. Basu, *Performance Analysis of Hadoop Map Reduce on Eucalyptus Private Cloud*, International Journal of Computer Applications, volume 79, pags 10-13, 2013.

[17] Apache Hadoop, *Hadoop 1.2.1 Documentation*, http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Data+Replication, (Access Date: 04 Dec., 2014).

[18] Amazon, *Amazon Elastic MapReduce (Amazon EMR)*, http://aws.amazon.com/elasticMapReduce/, (Access Date: 04 Dec., 2014).

[19] A. Iordache, C. Morin, N. Parlavantzas, E. FelleR, P. Riteau, *Resilin: Elastic MapReduce over Multiple Clouds Cluster*, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium on Digital Object Identifier: 10.1109/CCGrid.2013.48, 2013.

[20] C. He, D. Weitzel, D. Swanson, Y. Lu, *HOG: Distributed Hadoop MapReduce on the Grid*, SC Companion: High Performance Computing, Networking Storage and Analysis, 2012.

[21] G. N. Ramamoorthy. *K-Means Clustering Using Hadoop MapReduce. Published by UCD School of Computer Science and Informatics*.

[22] Apache Hadoop, *WordCount - Hadoop Wiki*, http://wiki.apache.org/hadoop/WordCount, (Access Date: 04 Dec., 2014).

[23] Project Gutenberg, *Free ebooks - Project Gutenberg*, https://www.gutenberg.org/, (Access Date: 04 Dec., 2014).

[24] O. O'Malley, *TeraByte Sort on Apache Hadoop*, Yahoo, 2008, http://sortbenchmark.org/YahooHadoop.pdf, (Access Date: 04 Dec., 2014).