Cloud computing for teaching and learning MPI with improved network communications

F. Gomez-Folgar, R. Valin, A. Garcia-Loureiro and T.F. Pena
Centro de Investigación en Tecnoloxías da Información (CITIUS)
University of Santiago de Compostela
Santiago de Compostela, Spain
(fernando.gomez.folgar, raul.valin, antonio.garcia.loureiro, tf.pena)@usc.es

I. Zablah
Sistema de Difusión de Radio y Televisión
Universidad Nacional Autónoma de Honduras
Tegucigalpa, Honduras
mrzablah@unah.tv

***Abstract*—Nowadays, the teaching-learning processes are being supported by the development of new technologies. During the recent past, technologies such as email, chat, audioconferencing, videoconferencing and webconferencing were incorporated as new tools in the teaching-learning process. Currently, another step is being walked with the development and the popularization of cloud technologies that are arousing great interest in educational environments. There has been an actively development of cloud platforms with the release of several open-source solutions to build private, public and hybrid clouds such as OpenNebula, Eucalyptus, OpenStack and CloudStack. Each of them has unique features that are not found in the others.**

In the most basic cloud service model, Infrastructure as a Service, it is possible to provide computational resources as virtual machines. In Computer Science this model offers to teachers and students the possibility of managing virtual infrastructures in which system administration and programming languages practices can be performed without compromising the configuration of the underlying physical compute nodes.

In order to test a cloud infrastructure as a tool for learning MPI, two different scenarios were evaluated in this work using CloudStack: a virtual cluster as a MPI execution environment, and an improved virtual cluster whose MPI communication latency was improved. The results of this study are presented in this paper.

***Keywords*-cloud; CloudStack; OpenMPI; Open-MX;**

I. Introduction

Cloud technologies [1]–[6] are arousing great interest in educational environments as well as in business companies [7], and they are emerging as new tools that can be employed to support teaching-learning processes in a similar way that, in the past, technologies such as email, chat, audioconferencing, videoconferencing, webconferencing, virtual classrooms, and collaboration suites were incorporated to support these processes. As a result, there is an increasing number of open-source solutions to build private, public and hybrid clouds. Some of the most popular platforms are OpenNebula [8], Eucalyptus [9], OpenStack [10] and CloudStack [11]. All of them have unique features that are not found on the others.

In the teaching-learning processes, clouds, under Infrastructure as a Service (IaaS) model, could be very useful due to the fact that cloud users usually employ virtualized resources. Hypervisors provide the necessary abstraction layer and isolation in the same way as a sandbox. As a result, virtualized learning environments allow us to use the computational power of the compute nodes without the need of changing the physical host configuration, reducing the systems administration effort and isolating the physical host configuration from the student's virtualized environment. Furthermore, they allow users installing different guests operating systems and testing software that can coexist under the same physical hosts without compromising or modifying its configuration.

In this work, two different scenarios of a cloud infrastructure based on CloudStack for MPI learning are introduced: a virtual cluster as a MPI execution environment and an improved virtual cluster whose MPI communication latency was reduced. This paper is organized as follows. In section II the architecture and characteristics of the CloudStack platform are presented. Section III describes the two teaching scenarios deployed under CloudStack for learning MPI. In the first one a basic deployment is described, whereas the second one describes a scenario with improved performance. Section IV describes the benchmarks to evaluate the performance of both scenarios. The results obtained are presented in section V. Finally, the conclusions of this paper are drawn in section VI.

II. CloudStack

CloudStack is an open-source cloud management platform, owned by Citrix, whose software architecture is shown in Fig. 1. It is composed by five types of components: Compute Nodes (CNs), Clusters, Pods, Availability Zones, and a Management Server. The Compute Nodes are hypervisor-enabled hosts that have installed and configured the CloudStack agent. These hosts are the basic physical block that allow us scaling the platform. Additional hosts can be added at any time to increase the provided capacity for guest Virtual Machines (VMs). The hosts are not visible to the end users, therefore, they can not determine which hosts have been assigned to them to execute their guest

VMs. A Cluster is a collection of CNs that share the same hypervisor type and have also access to the same Primary Storage system. The Primary Storage stores the root filesystem of guest VMs. Clusters are not visible to end users and represent the second level of scaling. A Pod is a collection of clusters. It represents the third level of physical scaling in the CloudStack platform. As clusters, Pods are not visible to end users. The Availability Zone is a collection of Pods and a Secondary Storage that stores predefined VM templates and ISO images. It represents the fourth level of physical scaling. The Availability Zones are visible to the end user who must select one of those to start a VM for the first time. The Management Server manages the entire cloud.

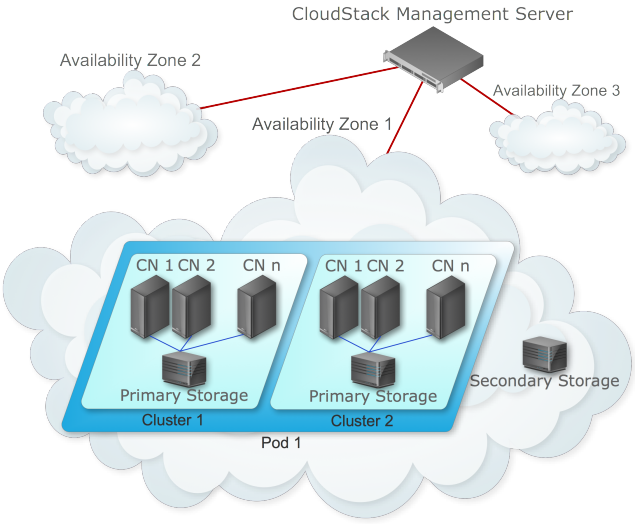

Figure 1. CloudStack architecture.

CloudStack supports three user roles: root administrator, domain administrator and not privileges users. The root administrator can manage the entire cloud. Domain administrators can perform the administrative operations for users who belong to that domain and do not have visibility into the physical CNs. The not privileged users can manage their own VMs.

The hypervisors supported by CloudStack are KVM [12], Citrix XenServer [13] and VMware vSphere [14].

This cloud platform can be managed completely through the Web management server. It also provides a RESTFUL API access to all its features. CloudStack also provides CloudBridge, which is a server process that runs in companion to CloudStack and provides an Amazon EC2 [15] compatible API to access to CloudStack using existing EC2-compatible tools. CloudBridge translates the EC2 API calls to the CloudStack's native API.

One of the most notable characteristics in CloudStack is the Web interface that provides a complete management of the cloud. We have observed very interesting options like the ability to define highly available VMs. They are kept operational by CloudStack without user or administrator intervention at all. Another interesting option for educational environments is the installation of an operating system using a standard ISO image. Its installation can be accomplished through the web interface without the need of using additional tools. This feature is very interesting because teachers or students can install their own VMs without the need of using predefined VM templates. Furthermore, teachers and students can create templates of their VMs that can be private, only visible for the users of a specific account, or public, visible for all users.

III. Teaching and Learning MPI with CloudStack

Message Passing Interface (MPI) is a language-independent communications protocol that has become a *de facto* standard for communication among processes that implements a parallel program using the message-passing model. Distributed memory supercomputer clusters often offers the use of MPI to their users.

The main goal of this article is to show how a cloud infrastructure can be used to teach MPI but following, at the same time, the Constructivism theory that allows students construct their own knowledge by means of their personal experience and interpretations. The role of the teacher is to be a help in the understanding, improving the learning quality and fostering the knowledge construction.

Students can deploy a safe infrastructure under CloudStack to learn the complete process including the installation of a VM, the configuration of the operating system, the installation of the MPI environment and the related development tools. Under this infrastructure, students must be able to carry out the performance analysis of their applications and testing and implementing different approaches to release a MPI solution for a given computational problem.

In order to test the CloudStack cloud infrastructure as a teaching-learning tool for MPI programming paradigm two scenarios were prepared and deployed. The first one is a basic testing setup scenario deployed as proof of concept, and the last one constituted an improvement from the first one, with the purpose of getting better performance. Both of them are described in the next subsections.

The purpose of these scenarios is to prepare the students for solving problems in complex environments. The Cloud technology will help to achieve this objective and thanks to the CloudStack Web interface, the teacher can assist the students easily and review their progress, focusing the attention on the most relevant topics. CloudStack provides flexibility to the teaching-learning process providing independence of time and space. The students can perform their activities without the need of being present in the computer laboratory.

The cloud infrastructure used is based on CloudStack 2.2.14 employing commodity hardware. CNs are Intel Core

i5 nodes with 8 GB of RAM, employing CentOS 6.3 64 bits as operating system, and KVM as the CloudStack managed hypervisor. The NFS server, acting as CloudStack Primary Storage, is a Core 2 DUO 6600 @ 2.40 GHz, with 4 GB of RAM, 500 GB hard disk (7200 RPM SATA) and CentOS 6.3 64 bit. The interconnection network of this cloud is an Ethernet Gigabit Network with a MTU of 1500 bytes.

A. *Basic scenario*

In order to test the CloudStack cloud infrastructure as a learning tool for the MPI programming paradigm, it is necessary to deploy a virtual cluster. A virtual cluster can be defined as a cluster composed by Virtual Machines (VMs) where the parallel applications are executed.

In computer science, a cluster is a group of interconnected computers that work together, and which can be viewed as a single system. Typically, as shown in Fig. 2, two types of components can be part of a cluster attending the way that they are used: head and nodes. The head, or master, is the computer where the users connect. The nodes are intended as computational resources that will be employed to run user applications. Typically, users do not have direct access to nodes so they cannot log in. Users will launch applications from the head that will be executed on the nodes. Each computer that compounds the cluster runs its own instance of an operating system.

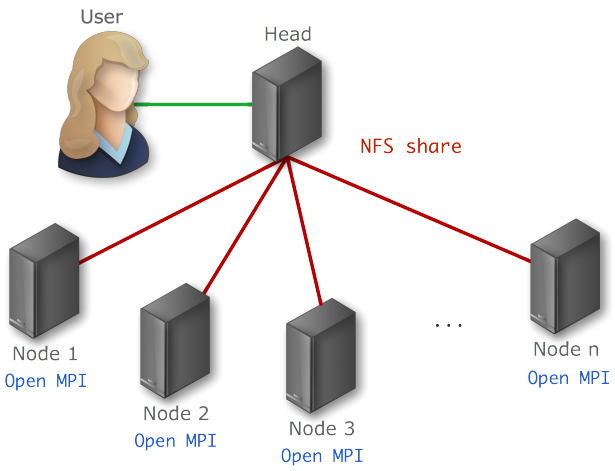

Figure 2. Cluster architecture.

The virtual cluster that students must deploy in CloudStack is composed by a VM configured as the head and two VMs configured as nodes. The deployed head is a VM employing a 10 GB hard disk, one core, 1 GB of RAM and CentOS 6.3 operating system. It also serves the home directory to the nodes that will compound the virtual cluster employing Network File System (NFS) as a distributed file system protocol.

The deployed nodes are VMs with a 10 GB hard disk, each one has one core CPU and 1 GB of RAM under CentOS 6.3 operating system. The nodes mount the head shared directories. A 1 Gb Ethernet network interconnection is being shared by the deployed virtual machines.

The deployed virtual cluster employs OpenMPI 1.6 [16] as MPI implementation. OpenMPI is an open source MPI-2 [17] implementation developed by a consortium composed by research, academic and industry partners. Its features include full MPI-2 standards conformance, thread safety and concurrency, dynamic process spawning, network and process fault tolerance, network heterogeneity support, and run-time instrumentation, among others.

B. *Improved scenario*

Due to the high latency of MPI communications over Ethernet networks using TCP, the performance obtained is limited. However, this latency can be reduced using Open-MX [18].

Open-MX is a high-performance implementation of the Myrinet Express message-passing stack over generic Ethernet networks. It implements the capabilities of the MX firmware running in Myri-10G NICs as a driver in the Linux kernel. For legacy applications, a user-space library exposes the MX interface to legacy applications. Open-MX supports Linux on any architecture and works at least on Linux kernels equal or greater than 2.6.15 version. It works on all Ethernet hardware that the Linux kernel supports and all connected peers, or compute nodes, must be on the same LAN. Therefore, any router can not be between them but switches. Open-MX is compatible with the IP traffic and can perfectly coexist on the same network and drivers. To setup Open-MX to be used by OpenMPI, it is necessary to take into account that OpenMPI must be compiled and installed enabling the Open-MX support.

The purpose of this scenario is to make students aware of the importance of the analysis of computer performance.

The virtual cluster employed in this setup has the same configuration as described previously but MPI communications are held by Open-MX, avoiding the overhead of TCP for communicating MPI processes.

IV. Benchmarks description

In order to test the scenarios described previously, and making students aware of the importance of performance evaluation, three types of applications were executed: Intel MPI Benchmarks [19], the HEAT_MPI [20] example, and the Gadget-2 [21] application. The first one, the Intel MPI Benchmarks 3.2.3 (IMB), provides a concise set of elementary MPI benchmark kernels. It has several program parameters such as message lengths or selection of communicators to run a specific benchmark. IMB also provides a standard and an optional configuration. If standard mode is used, all parameters mentioned previously are fixed and must not be

changed. The mode selected to test the virtual infrastructure is the standard ones. In this mode, message lengths varies from 0, 1, 2, 4, 8, 16 to 4194304 bytes.

The current version of IMB, contains different classes of benchmarks: Single Transfer, Parallel Transfer and Collective. The Single Transfer benchmarks are `PingPong` and `PingPing`. The Parallel Transfer benchmarks are `Exchange` and `Sendrecv`. The collective benchmarks are `Bcast`, `Allgather`, `Allgatherv`, `Alltoall`, `Alltoallv`, `Reduce`, `Reduce_scatter`, `Allreduce` and `Barrier`. `PingPong` is used for measuring startup and thoughput of a single message send between two processes. `PingPing` measures also the startup and throughput of single messages with the difference that messages are obstructed by oncoming messages. `Sendrecv` is based on `MPI_Sendrecv` and each process sends to its right and receives from its left neighbour in a chain. `Exchange` is a communication pattern often used in grid splitting algorithms, in which the group of processes is seen as a periodic chain, and each process exchanges data with both left and right neighbours in the chain. `Reduce` is the benchmark for the `MPI_reduce` function. It reduces a vector of length `L` float items employing the `MPI_SUM` operation. `Reduce_scatter` is the benchmark for the `MPI_Reduce_scatter` function that reduces a vector of length `L` float items employing the `MPI_SUM` operation. In the scatter stage, the `L` items are split as evenly as possible. `Allreduce` is the benchmark for the `MPI_Allreduce` function that reduces a vector of length `L` float items employing the `MPI_SUM` operation. `Allgather` is the benchmark for the `MPI_Allgather` function in which every process sends `r` bytes and receives a number of bytes that is equal to `r` multiplied by the number of processes. `Allgatherv` is the benchmark for the `MPI_Allgatherv` function that shows whether MPI produces overhead due to the more complicated situation as compared to `MPI_Allgather`. `Alltoall` is the benchmark for the `MPI_Alltoall` function in which every process inputs a number of bytes equal to `r` multiplied by the number of processes (`r` for each process) and receives a number of bytes equal to `r` multiplied by the number of processes (`r` from each process). `Alltoallv` is the benchmark for the `MPI_Alltoallv` function. `Bcast` is the benchmark for `MPI_Bcast` in which the root process broadcast `r` bytes to all. In this benchmark the root process of the operation is changed cyclically.

In the second place, we test the performance of the virtual cluster employing John Burkardt's HEAT_MPI [20], which is a C implementation of the 1D time Dependent Heat Equation employing a form of domain decomposition.

In the third place, we test the infrastructure employing the Gadget-2 software [21]. Gadget-2 is a freely available code for cosmological N-body/SPH simulations on parallel computers with distributed memory. It uses an explicit communication model implemented with the standardized MPI communication interface. Gadget-2 computes gravitational forces with a hierarchical tree algorithm and represents fluids by means of smoothed particle hydrodynamics (SPH). Gadget-2 can be used for studies of isolated systems, or in simulations that include the cosmological expansion of space, with or without periodic boundary conditions in both cases. In these types of simulations, Gadget-2 follows the evolution of a self-gravitating collisionless N-body system, and allows gas dynamics to be optionally included.

V. Results

This section shows the results for the three applications described in IV. They were obtained for the basic scenario, using TCP for communicating MPI processes, and for the improved scenario, using Open-MX for communicating MPI processes.

In the first place, we are going to show the obtained results for both TCP and Open-MX of the Intel MPI Benchmarks. The results obtained for both the `PingPong` and the `PingPing` single transfer benchmarks are depicted in Fig. 3. Notice that in all these figures the X-axis is in logarithmic scale. For the `PingPong` benchmark, the latency is reduced by around a 30% when Open-MX is used for communication in comparison to TCP, such as is shown in the square marks of the figure. The `PingPing` benchmark also gets its latency reduced, even in a bigger quantity (around a 36%) than `PingPong`, when Open-MX is used.

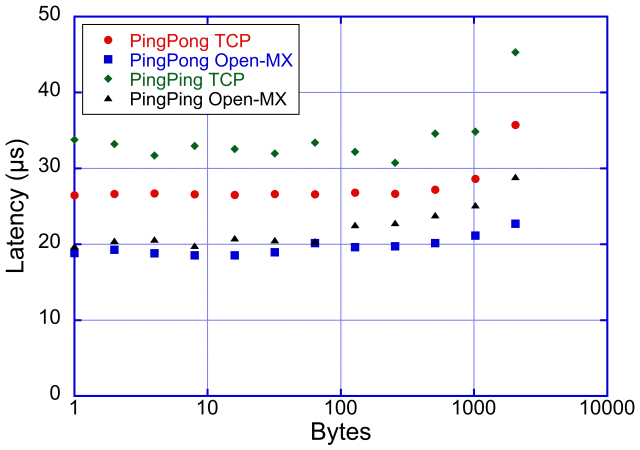

Figure 3. PingPong and PingPing single transfer benchmarks.

The results obtained for both the `Exchange` and the `Sendrecv` parallel transfer benchmarks are depicted in Fig. 4. For the `Exchange` benchmark, the latency is reduced by around 45% when Open-MX is used in comparison to TCP, such as is shown in the square marks of the figure.

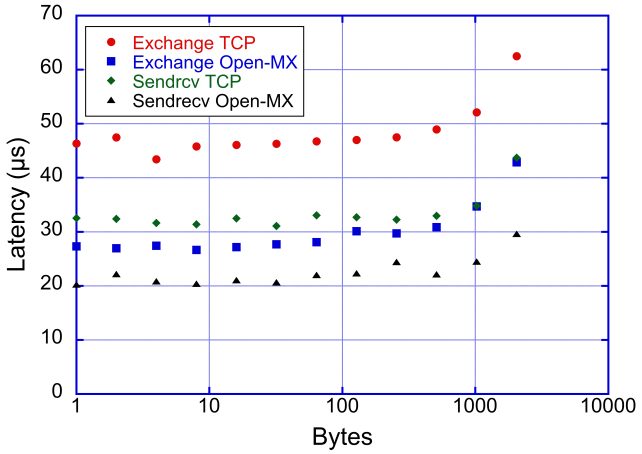

Figure 4. Exchange and Sendrecv parallel transfer benchmarks.

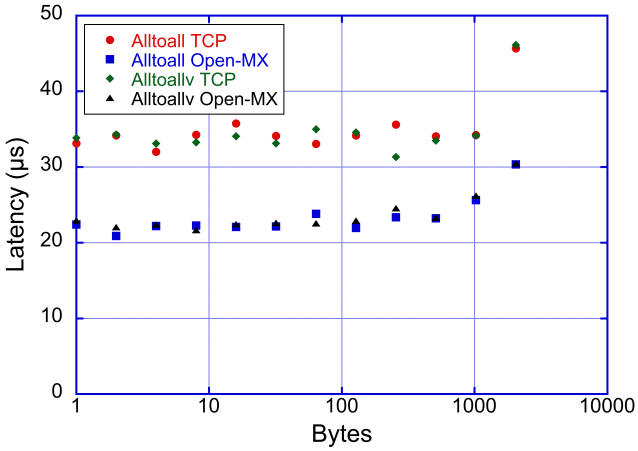

Figure 6. Alltoall and Alltoallv collective transfer benchmarks.

The `Sendrecv` benchmark also gets its latency reduced in 35% when Open-MX is used for communications.

The results obtained for both the `Allgather` and the `Allgatherv` collective benchmarks are depicted in Fig. 5. For the `Allgather` benchmark, the latency is reduced around 33% when Open-MX is used in comparison to TCP, such as is shown in the square marks of the figure. The `Allgatherv` benchmark also gets its latency reduced around 31% when Open-MX is used for communications.

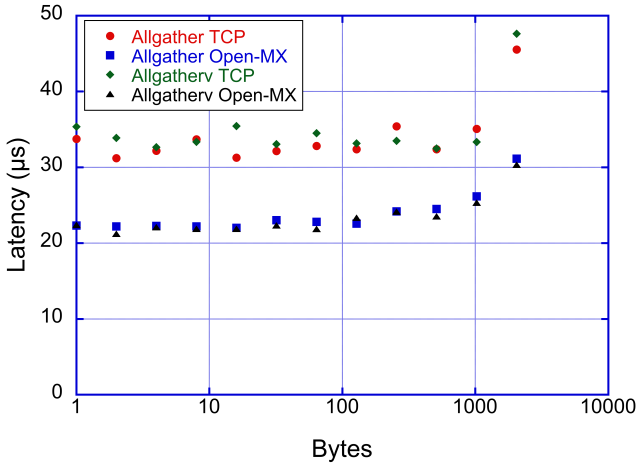

Figure 5. Allgather and Allgatherv collective transfer benchmarks.

The results obtained for both the `Alltoall` and the `Alltoallv` collective benchmarks are depicted in Fig. 6. For the `Alltoall` benchmark, the latency is reduced around 35% when Open-MX is used in comparison to TCP, such as is shown in the square marks of the figure. The `Alltoallv` benchmark also gets its latency reduced around 35% in a similar way as described previously when Open-MX is used for communications.

The results obtained for both the `Reduce` and the `Reduce_scatter` collective benchmarks are depicted in Fig. 7. For the `Reduce` benchmark, the latency is reduced around 35% when Open-MX is used in comparison to TCP, such as is shown in the square marks of the figure. The `Reduce_scatter` benchmark also gets its latency reduced around 32% when Open-MX is used for communications.

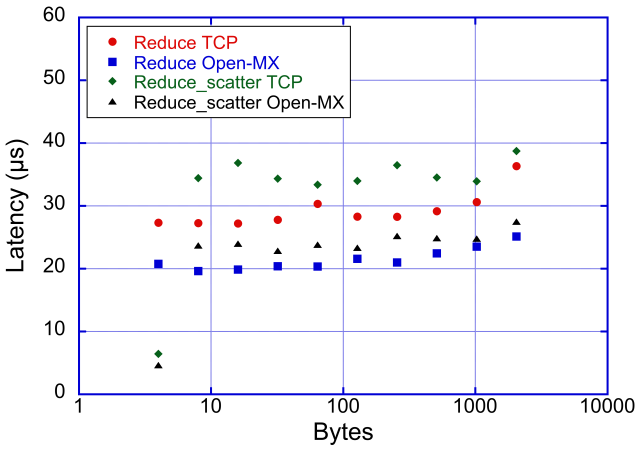

Figure 7. Reduce and Reduce_scatter collective transfer benchmarks.

In the second place, for the HEAT_MPI example, the performance is measured employing the computational elapsed time. When TCP is used to communicate MPI processes, employing two virtual nodes deployed under CloudStack, the elapsed time obtained was 22.708 milliseconds. When Open-MX is used, the elapsed time obtained was 15.878 milliseconds (a 30% better than TCP).

Finally, as a counter-example, Gadget-2 does not get a better performance when Open-MX is used for communicating MPI processes. This shows that the obtained improvements depend on a large extent of the problem.

VI. Conclusions

The development of cloud technologies are arousing great interest in educational environments. Cloud technologies are emerging as new tools to support teaching-learning processes in a similar way that, in the past, technologies such as email, chat, audioconferencing, videoconferencing, webconferencing, virtual classrooms or collaboration suites were incorporated to support these processes. Two characteristics of clouds are the use of virtualization technologies and the isolation between virtualized resources and the physical infrastructure. These characteristics convert this type of infrastructures into a very useful tool in teaching environments where teachers and students can perform experiments, avoiding compromising the configuration of the underlying physical infrastructure and reducing the effort of system administration tasks. As we have seen, teaching MPI on clouds, following the Constructivism theory, can be performed with the purpose of preparing the students for problem solving in complex environments. Therefore, two different scenarios, using commodity hardware and commodity interconnection networks, were deployed under CloudStack using KVM as hypervisor. The first one constitutes a virtual cluster for executing MPI applications. A virtual cluster can be defined as a cluster composed by virtual machines. The second one is an improved virtual cluster using Open-MX to get better latency of the MPI communications to make students aware of the importance of performing the computer performance analysis. Both virtual MPI infrastructures were tested employing the Intel MPI benchmarks, the HEAT_MPI example, and the Gadget-2 software. The obtained results show that executing MPI applications over the cloud were suitable and the latency of the MPI communications was reduced around a 30% using Open-MX in comparison to TCP. The elapsed time obtained for the HEAT_MPI example is also around a 30% better than TCP. However, depending on the implementation of the applications that will be executed using MPI, there are cases in which the improvements on the latency are not observed, as happened with Gadget-2. As it was shown, the experiments described can be performed by teachers or students as system administration, programming languages and performance measurement practices.

Acknowledgment

This work has been supported by FEDER funds and Xunta de Galicia under project 09TIC001CT, contracts 2010/28, and by Spanish Government (MCYT) under project TEC2010-17320.

References

[1] C. Babcock, *Management Strategies For The Cloud Revolution*, USA, 2010.

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

[3] B. Chee and C. Franklin, *Cloud Computing. Technologies and Strategies of the Ubiquitous Data Center*. Boca Raton: CRC Press, 2010.

[4] R. Krutz and R. Vines, *Cloud Security*. Indianapolis: Wiley Publishing, 2010.

[5] J. Rittinghouse and J. Ransome, *Cloud Computing: Implementation, Management, and Security*. Boca Raton: CRC Press, 2010.

[6] A. Velte, T. Velte, and R. Elsenpeter, *Cloud Computing: A Practical Approach*. USA: McGrawHill, 2010.

[7] K. Stanoevska-Slabeva, T. Wozniak, and S. Ristol, *Grid and Cloud Computing: A Business Perspective on Technology and Applications*. Germany: Springer, 2010.

[8] OpenNebula Project Leads. http://opennebula.org

[9] Eucalyptus. http://open.eucalyptus.com

[10] OpenStack. http://www.openstack.org

[11] CloudStack. http://www.cloud.com

[12] KVM. http://www.linux-kvm.org

[13] Xen. http://xen.org

[14] VMware. http://www.vmware.com

[15] Amazon Elastic Compute Cloud EC2. http://aws.amazon.com/ec2

[16] Open MPI. http://www.open-mpi.org

[17] MPIForum. http://www.mpi-forum.org

[18] OpenMX. http://open-mx.gforge.inria.fr

[19] Intel MPI Benchmarks: Users Guide and Methodology Description, Germany, 2006.

[20] MPI examples. http://people.sc.fsu.edu/ jburkardt/c_src/mpi/mpi.html

[21] Gadget. http://www.mpa-garching.mpg.de/gadget