EduCloud : a private cloud tool for academic environments

Paolo Cemim, Luis Carlos Jersak, Giuseppe Alves Lopes, Jair De Mello Junior and Tiago Ferreto
PPGCC-PUCRS
Email: {paolo.cemim, luis.jersak, giuseppe.lopes, jair.junior}@acad.pucrs.br, tiago.ferreto@pucrs.br

***Abstract*—Cloud computing emerged in the last years as a novel approach to deal with computational resources. The amount of resources available seems infinite, resources are charged per use (usually by the hour), provisioning or releasing resources is performed on demand and quickly. In addition, everyday more and more features are included by cloud providers in their cloud services menu. These novelties impose a shift in the traditional paradigm to build, manage and use computational resources. Therefore, there is a need to prepare current and future IT professionals to comprehend this new paradigm. This paper presents EduCloud, an academic private cloud tool that provides a simple environment to comprehend and experiment cloud computing concepts. EduCloud can be easily deployed using standard local computing resources, without the need of special hardware or access to external resources. It can be used to evaluate how a cloud computing environments works (e.g., cloud portal interaction, provisioning/decommissioning of resources, etc), and also to implement and experiment new strategies for cloud resources management (e.g., automatic scaling strategies, new billing models, etc) using standard computing resources.**

I. Introduction

It is common sense that Cloud Computing introduced significant changes in the manner IT infrastructure is used and provisioned. Instead of being a singular breakthrough, Cloud Computing is based on the composition of several innovations proposed in the last decades, such as virtualization, utility computing, IT infrastructure management, and others. Nevertheless, Cloud Computing is still being constantly redefined. Everyday, new proposals appear and disappear under the "Cloud computing umbrella" . Therefore, understanding deeply cloud computing is a continuous and laborious task, specially when evaluating its applicability in a real scenario. There are basically three approaches to understand and evaluate a cloud computing approach: (i) subscribing a public cloud provider, (ii) creating an on-premises private cloud, or (iii) creating experiments using cloud simulators.

Public cloud providers [1], [2] offer a production environment to execute elastic applications. All sorts of resources are provided to enable the execution of any type of application. Despite its robust and complete environment, they usually present a high learning curve. The amount of services provided usually increases its complexity and confuses the user when experiencing a cloud environment. Besides, most systems require the user commitment through a credit card which may not be feasible in some cases, such as in academia.

Another approach is deploying locally a private cloud [3]–[5]. There are several open-source packages to deploy production-grade private cloud environments. The installation is usually straightforward and, after installation, the system can be immediately used to host different types of applications. However, these packages usually present huge requirements, such as special processors with virtualization capability, huge amounts of memory and disk, and the need of a dedicated environment (usually as a hypervisor constraint). Performing trivial experiments using standard hardware is almost impractical. For instance, Eucalyptus requires at least 2 machines with 8GB of memory each, 8 2.4GHz cores for the frontend controller and 4 2.4 GHz cores for node controllers, and a Type 1 hypervisor which needs to run directly on top of host's hardware [1].

There are also cloud computing simulators [6]–[12] that can be used to learn the functioning of a cloud computing environment. These simulators provide abstractions for virtual machines, storage, networks, enabling the simulation of a whole Data Center and the behavior of applications running on it. However, depending on the utilization, the high abstraction can be prohibitive to evaluate a cloud computing environment, making the experience quite far from a real environment.

This paper presents EduCloud, a private cloud tool for academic environments. EduCloud enables the creation of a small cloud testbed using minimal requirements in terms of hardware and software. It can be effectively used to perform small experiments to evaluate common cloud computing characteristics, such as: automatic provisioning of virtual machines, interaction through a web portal, scaling of resources, among others. Due to its minimal requirements, it can be easily deployed in a classroom, using available resources (e.g. students' notebooks) to teach cloud computing characteristics. It can also be used to implement and evaluate new ideas to solve problems like resource allocation, billing strategies, etc.

This paper is organized as follows: In Section II we present and discuss relevant works related to our proposal. In Section III the Educloud platform is presented. In Section IV we analyze the platform and show the results obtained. Finally, Section V presents our conclusions and future works.

[1]Eucalyptus 3.1 Compatibility Matrix - http://www.eucalyptus.com/eucalyptus-cloud/iaas/compatibility

II. Related Work

In this section we present some relevant studies for the cloud computing area focusing on educational purposes. Currently, there are few options for teaching cloud computing concepts using a practical approach. Some of these options are private cloud platforms, some public cloud providers which offer free student accounts and simulators.

A. Public cloud platforms

Along with the growth of the cloud computing paradigm, several companies started to offer public cloud services. Some of these, like Amazon EC2 [1] and Microsoft Azure [2] have free trials and/or educational plans where students can use the cloud services the same way as regular paid plans. Although this provides a good experience as a user of cloud computing technologies, public clouds cannot provide in-depth knowledge about some concepts like the management and technical configuration of a cloud environment. For instance, a user of a public cloud is not allowed to change the scheduling policies or to implement a new algorithm for scale-up. Several other aspects like network configuration, consolidation policies and placement algorithms are also inaccessible. EduCloud assists in teaching not only the concepts from a user perspective, but also the technical concepts of a cloud computing environment from a "datacenter engineer" point of view, allowing the students to change parameters that alter the way the entire cloud runs.

B. Private cloud platforms

1) Open Nebula: Open Nebula [4] is an open source IaaS-type cloud development toolkit. It was projected for the deployment of private clouds, but using interface exposition it is possible to create public clouds. Also it is possible to create hybrid clouds through communication to other clouds. The Open-Nebula toolkit supports XEN, KVM and VMware hypervisors. The internal architecture of Open Nebula is divided in three layers: tools, drivers and core. The **tools** has several tools for managing the core through its interfaces. The available API allows developers to create custom tools. One of the tools distributed with the Open Nebula basic installation is the Command Line Interface (CLI), which allows administrators and users to manipulate the manual procedures of the virtual infrastructure. The **drivers** layer has a set of modules for communication with specific middleware. Some examples are the hypervisors, cloud services, file transfer mechanisms, etc. The **core** of Open Nebula is a set of components for managing virtual machines, virtual networks, storage, etc.

Like other private clouds, Open Nebula manages storage, network and host virtualization dynamically. It has three main functional components: the hypervisor, Virtual Infrastructure Manager and the Scheduler. The Hypervisor is the virtualization manager installed in each host for VM management. The Virtual Infrastructure Manager is the component responsible for centralized VM management and overall management of the cloud's resources. The Scheduler is responsible for several tasks like VM allocation policies, enforcement of restrictions, capacity reservations, etc.

2) Eucalyptus: Eucalyptus [3] is an open source framework for IaaS clouds. It supports XEN and KVM hypervisors. Eucalyptus' modules were projected with well-defined interfaces, aiming to facilitate the development of custom modules by its users. The Eucalyptus framework is composed of four main components:

Node Controller: The Node Controller is responsible for controlling the virtual machines on the host were it is running. This module responds the requisitions made by the Cluster Controller.

Cluster Controller: The Cluster Controller intermediates the requisitions from the Cloud Controller to the Node Controllers. It also manages the virtual network of its Node Controllers.

Storage Controller: The Storage Controller, also known as Walrus, is a storage service which allows users to store and retrieve data or images of VMs.

Cloud Controller: The Cloud Controller is the entry point for the Eucalyptus users. It is responsible for the general management of the cloud, making decisions and sending requests to the Cluster Controllers.

3) OpenStack: OpenStack [5] is an IaaS Project created by a partnership between Rackspace Cloud and NASA. It is free and open source under the Apache license terms. It is composed of three main components:

Nova (services infrastructure): Controls the lifecycle of the instances and manages the computational resources, network, authorizations and scalability. It has a REST based API and supports several hypervisors like XEN, XenServer, KVM, UML and Hyper-V.

Swift (storage infrastructure): Handles the cloud's storage tasks. It stores the virtual machines' images, provides redundancy, fault tolerance, backup services and archiving.

Glance (images infrastructure): It does the searching and retrieval of VMs' images. It can interact with the OpenStack native image system or with the Amazon's S3.

4) Platforms comparison: Table I shows a comparison of the presented platforms. The following characteristics were considered:

- Supported hypervisors;
- Platform's codebase;
- Security/authentication methods supported;
- Implantation models supported;
- Storage systems supported;
- Public cloud providers compatibility.

C. Simulators

1) iCanCloud: iCanCloud is a simulation platform aimed to model and simulate cloud computing systems. The main objectiveof iCanCloud is to predict the trade-offs between cost and performance of a given set of applications executed in a

TABLE I
CLOUD PLATFORMS COMPARISON.

	Eucalyptus	OpenNebula	OpenStack
Hypervisors	VMware, KVM, Xen	VMware, KVM, Xen	Xen, KVM, QEMU, LXC[1], VMWare[1], ESXi[1], Hyper-V
Code base	Java	Java, Ruby	Python
Security	Public/private keys	Passwords, SSH, RSA, LDAP	Security Groups
Implantation Models	Public[1], Private, Hybrid	Public, Private, Hybrid	Public, Private, Hybrid
Storage	Walrus	NFS, Secure CoPy	Swift
Public Providers	EC2 , S3	EC2	EC2, S3

specific hardware, and then provide to users useful information about such costs [10].

The iCanCloud simulator was written in C++ and has MPI support to enable simulations in distributed systems (not fully implemented yet). It has a graphical user interface to facilitate its usage and has models compatible with Amazon's EC2.

2) MDCsim: MDCsim [11] is an event driven simulator built on top of CSIM [7]. It is composed of three layers: the communication layer where all communications are modeled, the kernel layer which maintains a run queue for each CPU in the system, and the high-level application layer where processes like Web/Application/Database Servers are managed. The main drawback of MDCSim, as stated in [10] is that MDCsim is not available for public download as CSIM is a commercial product.

3) CloudSim: CloudSim [6] is a cloud simulator formerly based on GridSim [9], a grid simulator. In its later versions its core has been rewritten to not depend on GridSim anymore. It is written in Java, which is a drawback due to some JVM restrictions (for instance, 32 bit versions of the JVM can only use 2GB of system memory).

4) GreenCloud: GreenCloud [8] is a cloud simulator focused on energy efficiency, simulating the power consumption of nodes, switches, links and etc. It is based on the NS2 network simulator [12] and written in C++ language. GreenCloud mainly simulates the communication between processes in a cloud environment and allow the use of plugins to extend its functionalities.

In this section we presented some options to help teaching the cloud computing concepts using a practical approach. However, the options mentioned above have some drawbacks. Most of the private cloud platforms must be installed and deployed using dedicated hardware or present restrictions to aggregate heterogeneous hardware to its resource pool. Some public clouds (Amazon EC2 for instance) although providing free accounts, only provide the experience from a user point of view, not providing knowledge about the technical aspects of the environment. This complicates the teaching of some key concepts like scaling, placement or consolidation. Simulators are an inexpensive option but of difficult understanding and usage. Also, there are not many options of existing cloud computing simulators and some of these are not available for public access [11].

III. EDUCLOUD

The EduCloud is an open source platform for private cloud computing that can use dedicated or transient resources (like a student's laptop) to build its infrastructure. Transient resources do not need to be fully dedicated to run Educloud. The main purpose of Educloud is to provide a simple and user friendly private cloud infrastructure for academic environments. The service model implemented by EduCloud is Infrastructure as a Service (IaaS) and it uses VirtualBox [13] to provide virtualization. In the next sections the architecture of EduCloud and its functions will be explained.

A. Architecture

The architecture of EduCloud is divided in five main components: Cloud Controller, Node Controller, Centralized Storage, API Control and User Interface. Figure III-A shows the relation between the components of the architecture. These components will be explained in the next sections.

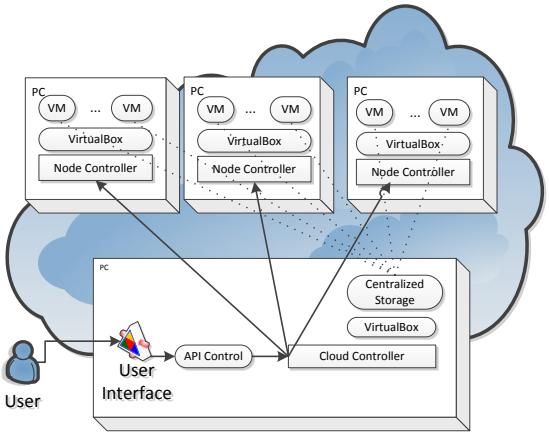

Fig. 1. Architecture of Educloud

[1]Not fully supported yet.

1) Cloud Controller: The Cloud Controller is the central component of EduCloud. It is responsible for managing all the routines of EduCloud, tasks running on the cloud and schedule new requests from the graphical interface. The use of a scheduler allows asynchronous task execution.

Also, the Cloud Controller manages the VM responsible for the centralized storage. This requires VirtualBox to be installed on the Cloud Controller. The management done by the Cloud Controller consists in creating LVMs (Logical Volume Manager) where the discs of the VMs provisioned by Educloud are stored. The Cloud Controller is also responsible for exporting the LVM to its respective VM.

The user management is another task of the Cloud Controller as well as its authentication. It also manages a database containing information about the structure and resources available and provides a communication interface through the control API. This communication interface sends the information required by the Node Controller to manage its requests.

2) Node Controller: The Node Controller has full control over one single host on the infrastructure. It is responsible for the execution of local tasks in its respective host. The local tasks done by the Node Controller are: creation and removal of virtual machines, starting up and shutting down virtual machines, registering new virtual machines and sending information about the virtual machines to the Cloud Controller. Each Node Controller needs VirtualBox installed and the communication between the Node Controller and VirtualBox is done through VirtualBox's WebServices. These tasks are better explained in Section III-B.

3) Control API: The control API is a Java component that abstracts the access to functions and information exported by the Cloud Controller's interface. The control API allows developers to integrate other functionalities or Java applications to Educloud.

4) Centralized Storage: The Centralized Storage Unit can be deployed by several ways, the most common being the utilization of a dedicated hardware called NAS storage. This type of equipment has high acquisition and management costs. To avoid this restriction, software-based solutions can be used. These solutions can emulate storage units on regular hardware or even in virtual machines. There are some ready to use solutions that emulate a Network-Attached Storage (NAS), such as FreeNAS [14] and OpenFiler [15], but some difficulties of integrating these solutions with Educloud forced the use of another option. In Educloud we used a distribution of Linux CentOS 6.2 with tools for exporting iSCSI and creation and management of LVMs.

The centralized storage unit consists of a virtual machine hosted at the same node as the Cloud Controller. This unit is managed by the Cloud Controller, and accessed by the Node Controllers to configure its virtual machines with disks hosted in it. The disk is remotely exported by the centralized storage to the VMs using iSCSI protocol. This way, each virtual machine has only one iSCSI target pointing to the centralized storage unit. The templates of virtual machines that can be provisioned by the Educloud are also stored in this unit.

The usage of a centralized storage area may cause a performance bottleneck on the provisioning of virtual machines since there will be dispute for resources during the creation of new VMs. But this is not a critical issue since the main focus of the EduCloud platform is to be deployed in small scale environments with educational purposes (e.g. a class room with a few computers) and not a high performance system.

5) User Interface: The User Interface is the component responsible for the user interaction with Educloud via a web browser. It provides access to all the Educloud's functionalities, allowing the inclusion, alteration and removal of tasks on the cloud, providing easy and fast usability of the Educloud system.

Also, the Educloud users access the portal using the user interface, where they must input an user login and password. There are two types of users: normal users and administrators. The administrators can manage the templates available in the cloud, the normal users and have access to the information about the nodes that compose the cloud infrastructure. The normal user has access to the information about its provisioned resources and can modify or allocate new resources.

B. Functions

In this section we present the functionalities of the Educloud platform. We show in detail the Template Registry, the Virtual Machine management and the scale-up and scale-out functions.

1) VM Template Registry: Each template loads a ready-to-use virtual machine image. This image must be already registered at Educloud. The template files must be previously saved inside the Cloud Controller's file system using the VirtualBox's virtual disk format (.vdi). The templates must be registered by an administrator through the User Intrface before they can be used by the users.

2) Virtual Machine Management: The main features of Educloud related to VM management are: Creation and Removal, Initialization and Shutdown, Scale-up and Scale-out. This features are described in the following sections.

Creation and Removal: The disk units management of each node is done by the centralized storage unit at the behest of the Cloud Controller. The Cloud Controller will register the new virtual machine at the Educloud's database, where it will remain in the "pending" state until the process is completed. The disks of the VMs provided by the cloud are kept by the Centralized Storage Unit inside a LVM. A copy of the template's disk is stored in this LVM. When this process is completed, the state of the virtual machine is changed to "finished" indicating it is ready for use. The process of removal of a virtual machine consists of deleting the information about the VM and the logical volume. This process ensures that all the data generated during the existence of the VM is erased.

Initialization and Shutdown: These tasks are requested by the Cloud Controller to the Node Controller through the VirtualBox's WebServices.

Scale-up: The scale-up process consists of request for resizing a virtual machine, allowing the alteration of the computational resources allocated for this machine. VirtualBox allows the dynamic resizing of some resources, without the need to reboot the virtual machine. However, some operating systems do not support dynamic hardware changes. In this case it is necessary to reboot the virtual machine. The resources that can be changed through scale-up are: amount of RAM, number of processors and processing capacity (CAP). The amount of RAM is the quantity of physical RAM allocated to the virtual machine based on the amount of RAM available on the node where the VM is being executed. The number of processors is based on the amount of cores available in the node. Also, as mentioned before, scale-up allows the alteration of the processing capacity of the VM. This parameter defines a percentage of processing capacity that can be used by the virtual machine. This way it is possible, for instance, to set a VM to use only 20% of the node's total processing capacity, allowing scale-up even when there are no cores available.

Scale-out: The feature of scale-out (or horizontal scaling) allows to increase or decrease the processing capacity of a virtual machine by duplicating or cloning it. The tool responsible for the load balance between the machines in Educloud is the LVS (Linux Virtual Server). Some LVS' characteristics relevant to this work are its versatility about hardware and software requirements, its free license policy and, most important, it is a native service of the Linux operating system. During the configuration of the LVS, the virtual machines related to a cluster are mapped into the LVS by its MAC addresses. Then the LVS manages the load balancing among these machines. To enable the scale-out feature it is necessary to inform that a virtual machine will use this feature as this means that, during the provisioning, an additional virtual machine must be created to act as the load balancer. The scale-out process consists of cloning a virtual machine and attaching it to the cluster through the LVS.

IV. Evaluation

To demonstrate the utilization of Educloud, some tests were performed. With such tests it is possible to evaluate the platform's performance, usability and operation. Initially we present the test environment followed by the analysis and results obtained.

A. Test Environment

The test environment consisted of three computers interconnected by a Gigabit Ethernet network. The operating system installed was Linux Ubuntu 11.10 and several other software tools required by the application, such as Java and VirtualBox, were installed.

The computers used in the test environment had two different hardware configurations. The machine responsible by the Cloud Controller had a 2.4Ghz Intel Core 2 Quad processor and 4 GB of RAM. The two computers responsible for the Node Controllers had 2.8Ghz Intel Core i3 processors with 4GB of RAM.

B. Results and Analysis

The tests performed focused on the main functionalities of the Educloud platform: Template Registration, VM Provisioning, VM Startup, VM Provisioning with scale-out, VM Startup with scale-out, Scale-out and scale-up. The results obtained are shown in Table II (further analyses are made in the following sections).

TABLE II
Test Results

Test	Time (secs)
Template Registration	5
VM Provisioning	255
VM Startup	6
VM Provisioning with Scale-out	453
VM Startup with Scale-out	25
Scale-out	258
Scale-up	1

1) Template Registration: To perform the Template Registration test we used a 2GB file that was already stored at the Cloud Controller. The Template Registration test total time was approximately 5 seconds. This time was performed on a centralized storage unit and the time is considered acceptable.

2) VM Provisioning: The greatest challenge during this test is its execution time. The process of template cloning makes a physical copy of the disk block informed by the used template, which is one of the most time consuming processes in this application. During this test an Ubuntu virtual machine is created, using a 2GB disk. As the storage unit is centralized, there is no need to transfer data through the network and the storage unit is responsible for the disk management. This is done via iSCSI protocol.

3) VM Startup: The centralization of the storage unit provides lower startup times as there is no need to send storage blocks through the network. This way it is only necessary to start the VM at the node controller and it will be ready to the user.

4) VM Provisioning with Scale-out: The entire execution process took approximately 453 seconds, representing a 77% increase in comparison to the creation of a virtual machine without the scale-out feature enabled. This is due to the fact that, in this case, it is necessary to instantiate to distinct virtual machines, one to act as the VM itself and another one for the Load Balancer, as seen in Section III-B2.

5) VM Startup with Scale-out: This process, besides being responsible for the VM startup, prepares the environment for the scale-out process. This way, the load balancer and the virtual machine are configured to support this functionality.

After this test was performed, it was found that this process takes approximately 25 seconds to complete, representing an increase when compared to the startup process of a regular VM. It is important to note that a VM with the scale-out feature enabled starts two virtual machines.

6) Scale-out: After the creation and initialization of the virtual machine and the configuration of the load balancer, the scale-out feature is available. This process is composed of two stages: initially a clone of the source VM is created and then this new instance is added to the existing load balancer. After this test was performed, we found that the process responsible to clone the VM takes almost the entire time to conclude the whole task (approximately 98%).

7) Scale-up: During the execution of the scale-up tests, three available parameters were tested showing this functionality. The tests were performed on a VM running Linux Ubuntu, configured to initially use 1 processor with a 30% load cap and 256MB of RAM. All the parameters were successfully changed and the VM had its computational resources reallocated instantly. The entire process was completed in less than 1 second mainly as a consequence of the Node Controller and VirtualBox integration. The Node Controller requests direct access to the VM and reallocates its resources. Although the resource reallocation was dynamic, the operational system was unable to identify it, requiring a VM reboot.

V. Conclusions and Future Work

In this paper we presented Educloud, a simple tool to build cloud computing testbeds using standard hardware and software. Educloud enables the execution of several tasks related to the management of a cloud infrastructure, serving as an option to demonstrate the concepts of cloud computing. Another feature of Educloud is the possibility of expanding its functionalities through its Control API or modifying its internal source code.

Considering that the goal of EduCloud is not to substitute other private clouds solutions, such as Open Nebula [4], Eucalyptus [3] or OpenStack [5] , performance was not the main issue when implementing the tool. However, results show that the time required to perform by the tasks is acceptable for an academic scenario. It is important to note that EduCloud enables the deployment of a private cloud using heterogeneous resources, composed by common hardware usually found in academic environments. This is one of the most important characteristics of Educloud.

As future work we intend to add support for several hypervisors, reduce the load balancer and centralized storage size and allow integration between private and public clouds. Also we plan to add new features, such as dynamic scaling and live migration. We provide more details about each future work below.

- **Multi-hypervisor** Supporting several hypervisors would make EduCloud more platform independent. Currently it supports only VirtualBox but other hypervisors could be integrated, like Xen [16] or VmWare [17].
- **Reduced load balancer, centralized storage and template sizes** It would reduce the disk capacity required by the Cloud Controller and Node Controller and also would decrease the size of the Educloud's package, facilitating its distribution to students.
- **Integration between private and public clouds** Integrating EduCloud with other private and public clouds could make Educloud's infrastructure more scalable, allowing to allocate more resources when available.
- **Dynamic Scaling feature** Implementing a dynamic scaling strategy would provide better utilization of the cloud's resources as the resources allocated would be only what the application running in the cloud demands.
- **Live Migration feature** Supporting live migration would solve the issue related to transient resources being part of the cloud. This feature would allow the live migration of VMs allocated in a node that will be removed from the cloud, to another free node in the infrastructure.

References

[1] "Amazon EC2," sep 2012. [Online]. Available: http://aws.amazon.com/ec2/

[2] "Windows Azure," sep 2012. [Online]. Available: http://www.windowsazure.com/

[3] "Eucalyptus.com," sep 2012. [Online]. Available: http://www.eucalyptus.com

[4] "OpenNebula.org," sep 2012. [Online]. Available: http://opennebula.org

[5] "OpenStack.org," sep 2012. [Online]. Available: http://www.openstack.org

[6] R. N. Calheiros, R. Ranjan, C. A. F. D. Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *CoRR*, vol. abs/0903.2525, 2009.

[7] "CSIM Development Toolkit," sep 2012. [Online]. Available: http://www.mesquite.com

[8] D. Kliazovich, P. Bouvry, and S. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, pp. 1–21, 2010.

[9] R. Buyya and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[10] A. Nunez, J. Vazquez-Poletti, A. Caminero, G. Castane, J. Carretero, and I. Llorente, "iCanCloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, pp. 185–209, 2012, 10.1007/s10723-012-9208-5.

[11] S.-H. Lim, B. Sharma, G. Nam, E.-K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform." in *CLUSTER*. IEEE, 2009, pp. 1–9.

[12] S. Mccanne, S. Floyd, and K. Fall, "NS2 (Network Simulator 2)," http://www-nrg.ee.lbl.gov/ns/. [Online]. Available: http://www-nrg.ee.lbl.gov/ns

[13] "Oracle VM VirtualBox," sep 2012. [Online]. Available: https://www.virtualbox.org

[14] "FreeNAS," sep 2012. [Online]. Available: http://www.freenas.org

[15] "OpenFiler," sep 2012. [Online]. Available: http://www.openfiler.com

[16] "Xen," sep 2012. [Online]. Available: https://www.xensource.com

[17] "VmWare," sep 2012. [Online]. Available: http://www.vmware.com