Modelling of virtualized servers

Ákos Kovács, Gábor Lencse

***Abstract*—The virtualized systems are one of the key elements of the next generation IT infrastructures. Modeling it will prevent mistakes, and oversized management. Opennebula is one of the most current open source cloud management solutions. Together with Haizea, they are a powerful tool to model and manage a virtualized infrastructure trough the API of Openebula. Haizea has multiple scheduler algorithms that are more effective than that of Opennebula. We performed several experiments to compare simulation and measurement capabilities of Haizea and we highlighted some difference between them which can be reduced if required.**

***Keywords*—cloud computing, Haizea, Opennebula, simulation,**

I. Introduction

Virtualized infrastructures are spreading around the world. They can optimize the performance resulting in lower TCO (Total Cost of Ownership) and greatly increased manageability of IT systems. The next evolution jump was the cloud computing systems. In this solution, the IT engineer only maintains the hardware, and the end users only rent the infrastructure. The most accepted definition of cloud computing was published by NIST [1] which defines the five essential characteristics of the cloud computing systems.

Modelling these system are one of the most researched topics. Many companies hosts virtual machines to sell them as a service. Predicting how many virtual machines can be operated using a given hardware infrastructure, or how much time it consumes to create a given number of virtual machines is very important to them. Opennebula [2] is a virtual infrastructure engine which can deploy, monitor and control virtual machines across many physical nodes. Haizea [2] was developed by the University of Chicago. It is an open source lease management architecture which can be used by Opennebula as a regular scheduler. With these two, they can manage physical nodes to automate the generation of virtual machines defined by templates. Haizea can also work as a virtual infrastructure simulator, which can predict (based on a model) how many virtual machines can be safely run in an infrastructure.

We simulated and analyzed a system built using a Bladecenter and Haizea in simulation mode as well as Opennebula mode to do experiments and compare them to each other. The remainder of this paper organized as follows. First, a brief introduction is given about the system, what kind of hardware was used for the experiments. Second, the modeling with Haizea is illustrated. Third, our experiments are described. Fourth, the results of our measurements are presented and discussed. Finally, our conclusions are given.

Manuscript received February 5, 2015.

Á. Kovács is with the Department of Telecommunications, Széchenyi István University, Egyetem tér 1. H-9026 Győr, Hungary (phone: +36-30-459-9026; fax: +36-96-613-646; e-mail: kovacs.akos@sze.hu).

G. Lencse is with the Department of Telecommunications, Széchenyi István University, Egyetem tér 1. H-9026 Győr, Hungary (e-mail: lencse@sze.hu).

II. Test Environment

An IBM Bladecenter was used as the test environment and VMware virtual machine was used as the cloud engine. The specifications were the following:

- cloud engine: VMware ESXi Virtual Machine (VM version 7) *2 CPUs, 2GB RAM 1x20GB Disks (iSCSI), 1x100GB (NFS), Debian 6.0.4 OS*
- cloud nodes: HS21 Bladeserver *2x L5240 Dual-core Xeon, 8GB DDRII ECC RAM, 73GB SAS Disk, 2x1Gb NIC, Debian 6.0.4 OS*

After the installation, we set up Opennebula on the cloud engine virtual machine. Its hardware requirements were minimal, it only consumed several MBs of disk space.
Opennebula supports a bunch of virtualization solutions, including VMware, Xen, and KVM [3].
We used KVM virtualization on the cloud nodes, with access to two networks. One for management and one for Internet access.

Opennebula uses remote command execution trough SSH tunnel, and for that we had to set up key based authentication between the cloud engine and the cloud nodes. For being able to manage the cloud nodes, we also had to set up the proper drivers for the virtualization solution we have chosen. These were the following:

- *im_kvm* (Information Manager): this driver gathers information about the cloud nodes e.g. numbers of running virtual machine and available memory
- *vmm_kvm* (Virtual Machine Manager): this driver monitors the virtual machines on the cloud nodes
- *tm_nfs* (Transfer Manager) this driver transfers the virtual machine images which are defined by the virtual machine template.

For the proper operation, we have to declare a virtual network in Opennebula with pairs of MAC addresses and IP addresses. With this, we are able to add a DHCP server with host directives to manage the Virtual Machines to get the right IP addresses. Opennebula provides shared storage to the cloud nodes, which we implemented by an NFS server on the cloud engine.

III. Modelling in Haizea

Haizea can be used as a scheduler instead of Openebula's default. Haizea supports advance reservation leases such as

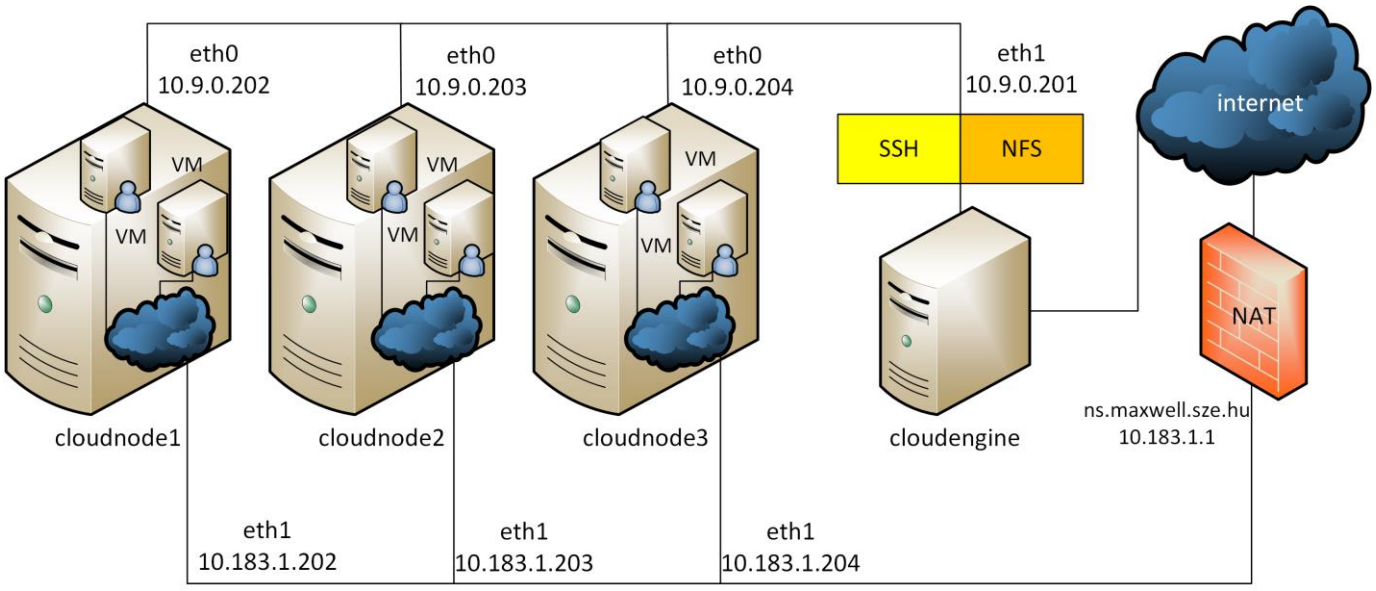

Fig. 1. Test system structure

best-effort leases [4] where the virtual resource is allocated as soon as it is available, or the request is placed in a queue when it is necessary (as no resource is available).Haizea leases contain various information which include the hardware and software resources and the time or availability when these resources can be accessed. Haizea commands are separated into three main blocks:

- *request block*: incoming requests, which can be added manually through CLI (Command Line Interface) or can be read from a special formatted XML file
- *scheduler block*: this block processes the requests which determine which virtual machine starts or stops and when
- *working block*: this block sends orders to the simulation (or in *openebula* mode to the Opennebula) to manage Virtual resources.

To run *Haizea* instead of the default scheduler of the Opennebula, the lease must contain the `Haizea` option in an Opennebula request. The simulation can be set up by two files. The first file contains the performance of the infrastructure and the second one contains the load of the system. The first file also has information about the transfer parameters of the virtual machine images (image size and transmission channel speed) which must be defined based on the real system [5].

In simulation mode, we have to define the resources for the Haizea. The most important ones are CPU and memory parameters and the number of cloud nodes. Also some other things has to be defined such as the clock is simulated or real time. We added four CPU cores for each of the nodes, and 7700 MB memory, because the Operating system which runs the KVM virtualization also uses some system memory from 8192MB.

```
[general]
loglevel: DEBUG
logfile: log/Haizea_sim_tilb_1.log
lease-failure-handling: exit-raise
mode: simulated
lease-preparation: imagetransfer

[scheduling]
policy-preemption: ar-preempts-everything
wakeup-interval: 10
suspension: none
migration: no

[simulation]
clock: simulated
starttime: 2013-04-04 11:03:15
resources: 3  CPU:400 Memory:7700
imagetransfer-bandwidth: 60
stop-when: all-leases-done

[tracefile]
tracefile: /srv/cloud/one/sims/sze_tilb_sim.lwf
```

Fig. 2. Haizea configuration file

After that, we created the XML file that describes the load of the system. The XML file must contains various information for example the amount of virtual resources (CPU, memory, and system image file), and the starting time of the lease. This file also describes the duration of the lease as well. We added all the leases into one file. We defined homogenous load for the simulation. All the virtual machines had 1 CPU core, 1 GB memory and 1 NIC. All the machines working with the same vanilla Debian image we created manually. The request of the virtual machines was sent at the 00:00:00 time. The duration of the virtual machine lifecycle was generally 1 hour for all of them. And the starting time was generated with Poisson distribution shown in Fig. 3. The request were overlapping with each other so it forced the Haizea to use best-effort algorithm.

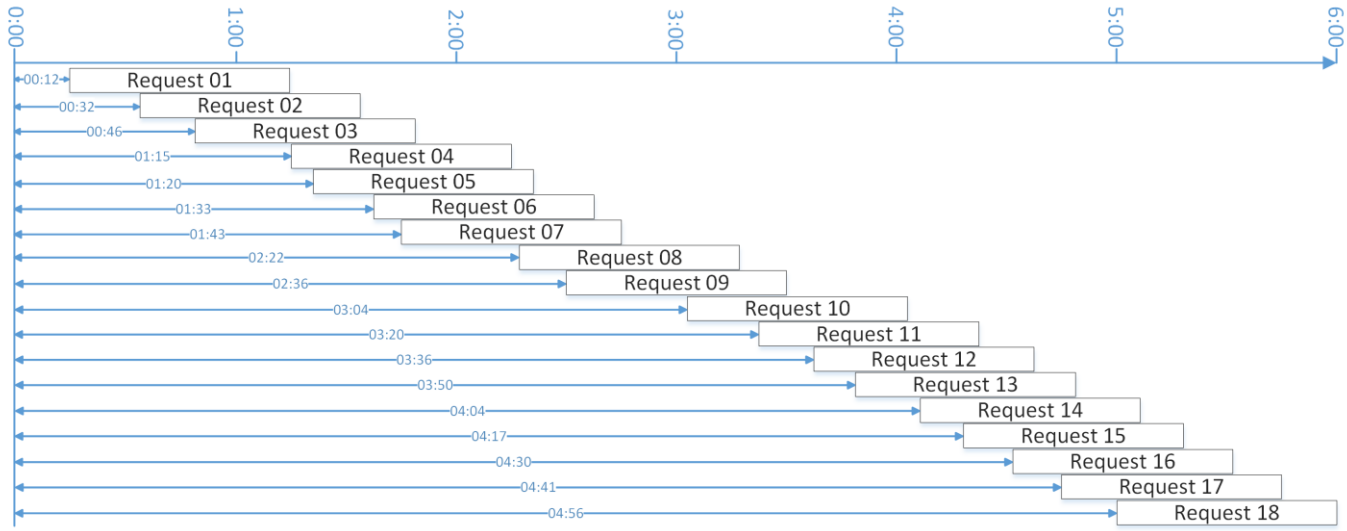

Fig. 3. The Poisson distribution of the requests

IV. MEASUREMENTS

A. *Testing the Starting of the Virtual Machines*

For the measuring mode, we generated the same jobs as for the simulation. We created 18 virtual machine description files with the same parameters as we defined in the simulation, only the virtual machine names and the starting times were different in each file. We used a simple bash script to run the measurement. It is important to notice that whereas the simulation finished almost instantly, the execution of the bash script took 5 seconds in average. Because the whole measurement took about 6 hours this difference was negligible. In measurement mode we used Haizea to schedule the virtual machines and Opennebula for deploying them. Only a few changes had to be made in the configuration file because the available resources were given by Opennebula and not manually and of course we had to define the Opennebula host, which one, in our case, was the same machine that executed Haizea.

After we started the measurement, the Opennebula was filled up with the requests, and at their starting time it copied the given virtual machine image file to a separate directory. After that it generated the virtual machine definition file and finally the virtual machine booted with the given parameters.

After the experiments we used some Linux based text processing tools (*sed*, *awk*, *grep*) to process the log files for producing the results.

We did not deal with the stopping time of the virtual machines, because we modified the shutdown script not to shut down but delete the virtual machines for simplifying the experiments.

TABLE I
DIFFERENCE BETWEEN SIMULATION AND MEASUREMENT

Haizea	Opennebula	Difference
(hh:mm:ss)		
8:29:09	8:33:17	0:04:08
8:49:09	8:52:44	0:03:35
9:03:09	9:07:21	0:04:12
9:22:10	9:26:17	0:04:07
9:37:10	9:41:38	0:04:28
9:50:10	9:53:38	0:03:28
10:00:11	10:04:00	0:03:49
10:21:11	10:24:46	0:03:35
10:39:11	10:43:43	0:04:32
10:53:12	10:56:47	0:03:35
11:21:12	11:25:17	0:04:05
11:37:12	11:40:55	0:03:43
11:53:12	11:56:54	0:03:42
12:07:13	12:10:49	0:03:36
12:21:13	12:24:47	0:03:34
12:34:13	12:37:47	0:03:34
12:47:14	12:51:27	0:04:13
12:58:14	13:01:56	0:03:42
13:13:14	13:16:45	0:03:31
	Average:	**0:03:51**
	Std. deviation:	**0:00:20**

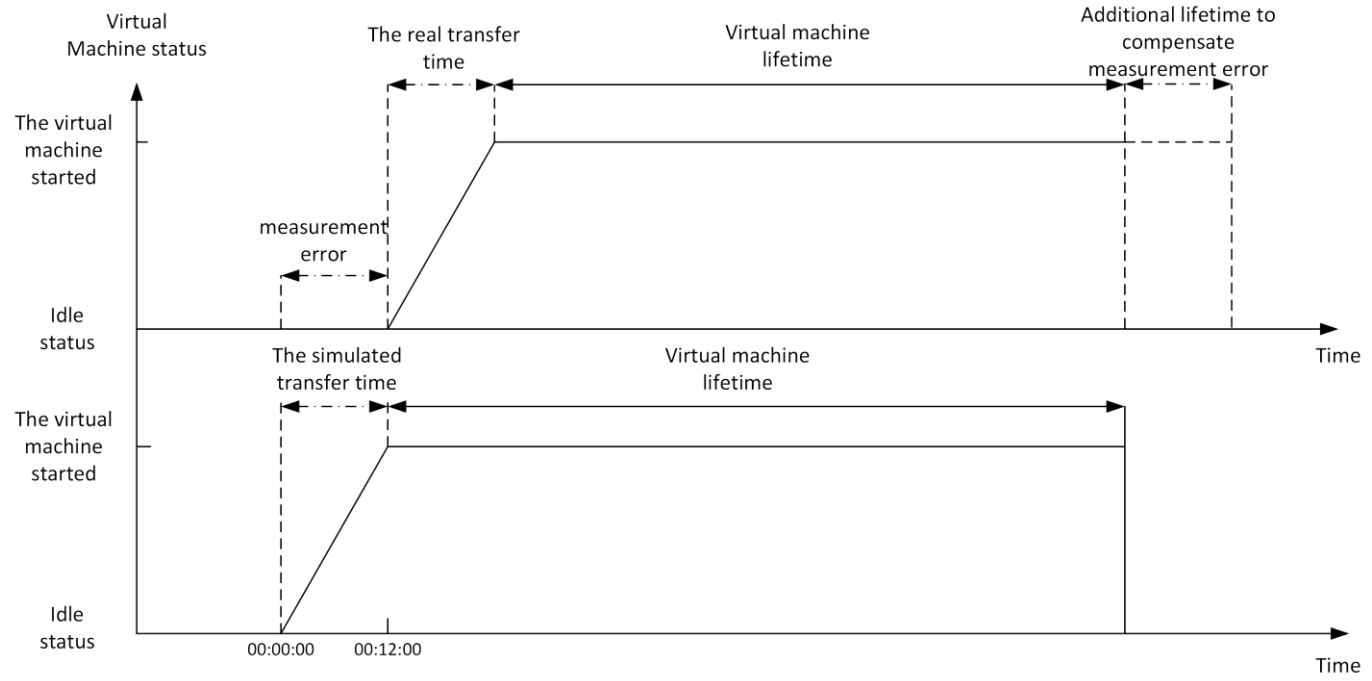

Fig 4. The static difference between the simulation and the measurement

As we can see in Table I there is a static difference between the simulation and the measurement with an average of 3 minutes and 51 seconds. The generated image file size was 4096 MB. It took averagely this time to clone an image file to the predefined place.

B. Examining the Limits of the System

We examined the available recourses both in simulation mode and in measurement mode. We generated a special Virtual machine with only a 40 MB system image, to decrease the deploying load of the system and minimalize the static delay. Based on the parameters we defined in the previous simulation it took about 3 seconds to deploy such a tiny Virtual Machine. We defined a script that generated 24 requests with homogenous 1 GB memory allocations and 1 CPU. In Linux system, a quad-core CPU is indicated as 400% CPU, whereas in Haizea all the CPUs are specified in percentage but only scaled up to 100%. To be sure that Haizea simulation and measurement uses only one core per Virtual Machine we had to define one core as 25% of the maximum CPU available in one host.

We got the same results in the simulation and in the measurement. The system could not generate the 24th Virtual Machine because the lack of available memory. In the simulation, we defined 7700MB memory per host total of 23.1 GB memory. It cloud simulate only 23 Virtual Machines the same as the measurement result.

C. Limits with Maximum Load

To test the stability of the system, we repeated the experiment with adding high CPU consuming script. To minimalize system image size, we used a simple script that copies random numbers from `/dev/urandom` to `/dev/null`. This script generated high CPU usage without charging the I/O subsystem.

The high load of the system did not cause significant difference in the starting times. Whereas in the experiment without high CPU usage the deploying time was about 3 seconds, in the experiment with high CPU usage it took about 4 seconds per Virtual Machine.

V. DISCUSSION OF THE RESULTS

Haizea can manage two cloning techniques. The first called *image preparation* which means that we define the moment when the virtual machine must be reachable and usable. In that case, Haizea uses the transfer bandwidth which is defined in the configuration file to calculate how much time it takes to transfer image files depending on the size. The second one called *unmanaged* when we only could define the moment when Opennebula starts to clone the virtual machine image file. Unfortunately Haizea supports *image preparation* only in simulation mode but not in Opennebula mode [6]. This the reason of the static delay between simulation and the measurement results.

As Fig. 4 shows we could compensate the Virtual machine lifetime in measurement mode, to manage the exact 1 hour lifetime. However the knowledge of the time necessary for the image transfer is a prerequisite for this kind of compensation.

In some cases it is not a problem that a virtual machine deployment is delayed a few minutes. For example, when a virtual machine will be a productive unit of a system and we does not want to delete in the near future. But in some special cases it is necessary to deploy Virtual Machines in time. For example, a lesson must be started exactly at a predefined time in a school. It is unacceptable to delay a it because of IT infrastructure.

VI. Recommendation for the Development of Haizea

We miss some parameters from Haizea which should be implemented in a future version. First, the image preparation feature in Opennebula mode for better usage. Second, in simulation mode we can't define standard deviation for the deploying. It is unequivocal that when we copy a system image with the size of 4 GB about 30-40 times its copying time will not take exactly the same time.

VII. Conclusion

We have demonstrated that deployment of the images causes a static difference between the starting times of the simulation and the Opennebula mode results.

We have shown that one can minimize the image deployment time for example with tiny images and using a remote file system to store non system files therefore the difference between the simulation and the Opennebula mode can be efficiently reduced.

We have shown that the high CPU load caused no significant difference in the starting time of the Virtual Machines.

References

[1] P. Mell, T. Grance, *The NIST Definition of Cloud Computing*, NIST Special Publication 800-145, (September 2011) http://dl.acm.org /citation.cfm?id=2206223

[2] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds", *IEEE Internet Computing*, vol. 13, no. 5, pp. 14-22, (Sep.-Oct. 2009), DOI: 10.1109/MIC.2009.119.

[3] OpenNebula.org, [Online] http://www.opennebula.org

[4] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster. "Capacity Leasing in Cloud Systems using the OpenNebula Engine", in *Workshop on Cloud Computing and its Applications 2008 (CCA08)* October 22-23, 2008, Chicago, Illinois, USA

[5] Haizea, [Online] http://haizea.cs.uchicago.edu

[6] Borja Sotomayor, "[Haizea] image transfer not working in Haizea 1.0 + Opennebula 1.4" [Online] https://mailman.cs.uchicago.edu/pipermail /haizea/2011-April/000307.html