New approaches for an (almost) automatic set-up of elastic cloud infrastructures

Pellegrino J., Amoroso A., Astorino F., Bagnasco S., Bianchi F., Destefanis M., Maggiora M., Yan L.

102° Congresso della Società Italiana di Fisica, Padova, 26/09/2016

Jacopo Pellegrino - jacopo.pellegrino@infn.it
Introduction
Introduction

The cloud computing paradigm provided access to a shared pool of computational and storage resources.

Resources are easily available to users.

Setting a cloud infrastructure up may not be trivial by the providers’ point of view.
Introduction

Cloud computing is widely adopted for HEP purposes.

The goal for

- **users** is to be able to access computational resources when needed.
- **providers** is to maximize the efficiency of the cloud infrastructure.
Introduction

Our approach addresses two common issues of cloud infrastructures:

- **Usability**: simplifying the set-up and installation process.
- **Efficiency**: making the usage of resources more dynamic, flexible and efficient.

The aim is to improve existing cloud infrastructures and give sites with limited knowledge easier access to cloud technologies.
Automatic set-up
Automatic set-up

**Aim:** setting up a **OpenNebula** hypervisor minimizing the user interaction during the installation process.

Installation performed via usb key.

**Hardware used for test:**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Dell Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2 x Intel(r) Xeon(r) E5-2650 v3 @ 2.30 GHz</td>
</tr>
<tr>
<td>Cores</td>
<td>20 physical, 40 hyper threading</td>
</tr>
<tr>
<td>RAM (GB)</td>
<td>160</td>
</tr>
</tbody>
</table>
Automatic set-up - network topology

- Server
- Storage
- Private network
- Public network
- Internet
- Firewall
- Virtual Machines (vm1, vmN)
- Network Components (em1, em2)
Automatic set-up

➔ Create kickstart file
➔ Prepare customized ISO
➔ Make bootable usb drive
➔ Install on server
➔ Test the installation
A **kickstart** file contains the information needed to perform the installation in order to *avoid the user providing them*. Some parameters may be left to user input (network parameters, keyboard layout, ...).

It is possible to specify which **packages or additional software** have to be installed.

Software installed via kickstart: OpenNebula, squid proxy, and rOCCI.
Automatic set-up - customized ISO

Starting from a kickstart and a standard iso it is possible to modify the iso so that it will look for the given kickstart at installation time.

The standard iso adopted during the test is: CentOS- 6.7-x86_64-netinstall.iso

Pay attention to the location of the kickstart: a good idea is to refer to the usb drive using its “label”.
Automatic set-up - bootable usb drive

Format (FAT32, mbr) a usb drive (2GB is enough).

It's important to give it the **name** indicated in the customized iso creation process. Otherwise the kickstart will not be found when the installation begins.

Make the usb bootable either via command line or using an application such as **Unetbootin**.
Automatic set-up - install on server

Start or reboot the machine and plug in the bootable usb drive, choose the **boot from usb drive** option.

The installation will ask the user to provide the parameters not specified within the kickstart file, then the installation will proceed autonomously. For instance, the network parameters have been left to the user input.

The machine will reboot when the installation is over. Remove the usb drive and let the machine boot normally.
Automatic set-up - test the installation

Once the machine is up and running, users may control that the parameters have been properly set (network interface is up, disks are mounted, ...).

To verify that OpenNebula is working check the hypervisor status via the **Sunstone** interface.

The entire installation process requires about 1 hour (on the hardware used for the test).
Automatic set-up - test the installation

Test the installation running a customized VM to check that all the installed software run properly.

Create a **new image** in OpenNebula: a file .one will be provided with the proper path.

Create a **new template** in OpenNebula: a file .txt will be provided, insert image and network number.

**Instantiate a new VM**: the machine is ready in less than 1 minute and can be used, software mounted via CVMFS used in combination with the squid proxy.
Automatic set-up - about VMs

There are no constraints on OS: SL5, SL6, Ubuntu 14, ...

Strict requirements on **storage space**: the VM image has to be copied over the net each time a new VM is instantiated. It could be a bottleneck.

- **QCOW2** image format: Dynamic increase of the storage

- **Minimal OS** installation: Only the required software is installed
Elasticity of the infrastructure
Elasticity of the infrastructure

**Aim:** provide inter-experiment elasticity.

The CI is used by different stakeholders, the aim is to dynamically change quotas.

EXP2 has 100 cores
EXP4 has 200 cores
Together they have 300

The aim is to manage these 300 cores **dynamically** for EXP2 and EXP4.
Elasticity of the infrastructure

Directly change quotas may be not safe (different stakeholders are involved)

A “private” OpenNebula is introduced
The “public” OpenNebula only knows that EXP2 and EXP4 have 300 cores.

The “private” Opennebula can dynamically change quotas between EXP2 and EPX4 according to actual needs.

Variations of quotas can be performed via rOCCI.
Automatic set-up - about VMs

So far both the “private” and “public” OpenNebula have been set-up for testing purposes.

Interaction between them is a work-in-progress.
Future work
Future work

- Make “private” and “public” OpenNebula interact.

- Implement an algorithm for variation of quotas.

- Test submitting jobs.
Acknowledgements
Acknowledgements

This work has been supported by BESIII and Belle2.

Special thanks to my colleagues Antonio Amoroso\textsuperscript{1,2}, Flavio Astorino\textsuperscript{1}, Stefano Bagnaco\textsuperscript{2}, Fabrizio Bianchi\textsuperscript{1,2}, Marco Destefanis\textsuperscript{1,2}, Yan Liang\textsuperscript{2} and my supervisor Marco Maggiora\textsuperscript{1,2}

\textsuperscript{1}Università degli Studi di Torino, \textsuperscript{2}INFN, Sezione di Torino
Contacts

Jacopo Pellegrino
Università degli Studi di Torino, Dipartimento di Fisica

tel: +39 011 670 7478
mail: jacopo.pellegrino@infn.it
Backup

Jacopo Pellegrino
Università degli Studi di Torino, Dipartimento di Fisica

tel: +39 011 670 7478
mail: jacopo.pellegrino@infn.it
OpenNebula is a widely adopted, opensource cloud platform.

- **Simplicity**
  You do not need an army of administrators to build and maintain your cloud

- **Openness**
  You will run production-ready software that is fully open-source without proprietary extensions that lock you in

- **Reliability**
  Your cloud will run for years with little maintain

- **Flexibility**
  You can easily build a cloud to fit into your data center and policies

[Image from http://opennebula.org/about/why/]
More on cloud computing

- **The amount of resources** and the variety of applications is steadily increasing, manpower unfortunately is not.

- It is becoming almost mandatory to consolidate such resources to **achieve scalability** and economies-of-scale
  - Separate application management from infrastructure management.
  - Our Data Centers need to become providers of computing and storage resources, not (only) of high level services.

- The **Cloud approach (IaaS)** might help to better provision resources to the different scientific computing applications
  - Grid Sites, small or medium computing farms, single users,…
  - Admit dynamic resource relocation to increase CPU power for a Grid and reduce some other that are not using resources or having less priority.

- Several cloud computing projects are starting at national and European level
  - From definition of best practices and reference configurations to deployment of large-scale distributed infrastructures.
  - A local working cloud infrastructure will also allow to take immediately part in such activities.
More on cloud computing

- **On-demand self-service.**
  A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

- **Broad network access.**
  Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client.

- **Resource pooling.**
  Computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

- **Rapid elasticity.**
  Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand.

- **Measured service.**
  Cloud systems automatically control and optimize resource use by leveraging a metering capability at a level of abstraction appropriate to the type of service.
More on cloud computing

Service models:

- **IaaS**: *Infrastructure as a Service*.
- **PaaS**: *Platform as a Service*.
- **SaaS**: *Software as a Service*
More on cloud computing

- **IaaS**
  - Applications
  - Data
  - Runtime
  - Middleware
  - O.S.
  - Virtualization
  - Servers
  - Storage
  - Networking

- **PaaS**
  - Applications
  - Data
  - Runtime
  - Middleware
  - O.S.
  - Virtualization
  - Servers
  - Storage
  - Networking

- **SaaS**
  - Applications
  - Data
  - Runtime
  - Middleware
  - O.S.
  - Virtualization
  - Servers
  - Storage
  - Networking
Accessing experiment software

Access specific software for the experiment:

- install while preparing VM
  
  \textit{requires a lot of space}

- install during contextualization

  \textit{requires space and time}

- mount via \textbf{CVMFS}

  \textit{requires less time, install only required software}
rOCCI

**OCCI** (the *Open Cloud Computing Interface*) is a standard by the *Open Grid Forum*, specifying a protocol and API to perform various remote management tasks in clouds.

The rOCCI-server extends cloud managers, which are not OCCI-compliant natively, with its own OCCI interface. It is based on the rOCCI (Ruby OCCI) Framework.
Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages.

Squid has extensive access controls and makes a great server accelerator. It runs on most available operating systems, including Windows and is licensed under the GNU GPL.