A modular (almost) automatic set-up for elastic multi-tenants cloud (micro)infrastructures

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 082031 doi:10.1088/1742-6596/898/8/082031

A modular (almost) automatic set-up for elastic multi-tenants cloud (micro)infrastructures

A Amoroso[1,2], F Astorino[1], S Bagnasco[2], N A Balashov[3], F Bianchi[1,2], M Destefanis[1,2], S Lusso[2], M Maggiora[1,2], J Pellegrino[1,2], L Yan[2], T Yan[4], X Zhang[4] and X Zhao[4]

[1] Department of Physics, University of Turin
[2] INFN-Turin
[3] Joint Institute for Nuclear Research (Dubna)
[4] Institute of High Energy Physics, Chinese Academy of Sciences (Beijing)

E-mail: marco.maggiora@to.infn.it

Abstract. An auto-installing tool on an usb drive can allow for a quick and easy automatic deployment of OpenNebula-based cloud infrastructures remotely managed by a central VMDIRAC instance. A single team, in the main site of an HEP Collaboration or elsewhere, can manage and run a relatively large network of federated (micro-)cloud infrastructures, making an highly dynamic and elastic use of computing resources. Exploiting such an approach can lead to modular systems of cloud-bursting infrastructures addressing complex real-life scenarios.

1. Introduction

Cloud computing is perfectly suited to perform a strong separation between application and infrastructure, which is achieved by means of the virtualization approach.

A cloud computing infrastructure should be able to offer on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Among them, the resource pooling is one of the most powerful peculiarities for High Energy Physics (HEP), but is of more and more interest for Small Medium Enterprises (SME) as well. This feature should allow us to use different machines depending on our needs, leading thus to a system of shared resources, increasing the flexibility of the chosen model.

The service models which can be provided by a computing center could vary from case to case. They range from Software-as-a-Service (SaaS) to Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Generally, a cloud computing solution offers a IaaS service, and the approach described herewith profits of the IaaS recent developments performed within the BESIII Collaboration at IHEP [1], at JINR [2] and at the Computing Center of the INFN Section of Turin [3]; within the BELLEII Collaboration at KEK [4]. Nevertheless, the system we propose in this work can provide a centralized SaaS service. Indeed, it will be possible to skip the software customization part and directly use the provided machines.

We propose a cloud infrastructure management specifically designed to be simple and easy-to-use, that can be installed by reduced manpower with limited pre-existing cloud-specific skills by means of an auto-installing tool on a bootable usb drive, and that can be interfaced and managed

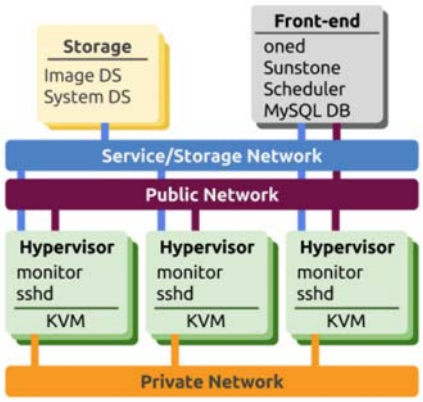

Figure 1. Network and virtualization schemes of OpenNebula.

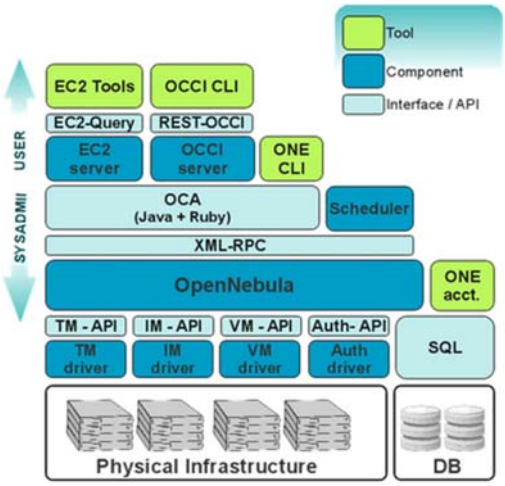

Figure 2. The OpenNebula stack.

by a central remote site. An high level of elasticity is provided, together with a straightforward dissemination of updates, contextualizations and optimizations.

A new modular approach exploiting our tool and cloud bursting among nested cloud infrastructures can offer new degrees of freedom to real-life scenarios.

2. OpenNebula

The automatic tools we developed deploy a Cloud Infrastructure (CI) based on OpenNebula [5] (ONE) and composed of physical hosts, often called Hypervisors (like the software), hosting the Virtual Machines (VMs). The virtualization, in our approach, is provided by KVM (ONE supports also vCenter and Xen). The Hypervisors host the monitoring daemons and expose *sshd* for system connection (Fig. 1).

The network structure is composed of three separate networks:

(i) a network used by OpenNebula and the infrastructure:
  (a) service and storage network: delivering monitoring and control information, besides allowing for image transfers;

(ii) two different networks used by VMs:
  (b) private network: exploiting private IPs for intra-cloud communications;
  (c) public network: exposing public IPs for incoming connectivity to VMs.

ONE allows in fact the creation of Virtual Networks by mapping them on top of the physical ones, and includes a Virtual Router appliance to provide network services (DHCP, DNS, etc.).

Resources and components are defined in ONE through Templates. All Templates share the same syntax, the most important ones being the VM Templates. A VM Template is the VM definition. An image is the VM disk image. A VM is an instantiated VM Template, and represents one life-cycle; several VMs can be created from a single VM Template.

The storage of the infrastructure is composed of different Datastores, where a Datastore is any storage medium used to store disk images for VMs. Typically, a Datastore will be backed by SAN/NAS servers, and must be accessible through the front-end using any suitable technology: NAS, SAN or direct attached storage. Service Datastores don't necessarily need to be shared across VMs: images can be transferred to the Hypervisors' disks through *ssh* and instantiated

locally. An Image Repository Datastore holds the OS images, while a System Datastore holds the running instances. In case the infrastructure includes a shared filesystem (FS), VMs can be live-migrated. Disk images can be persistent or volatile: the former ones saved back to the repository when a VM is destroyed, while the latter ones are lost with their changes.

The control node runs the ONE stack (Fig. 2), providing:

- oned (the main daemon);
- schedd (the VM scheduler);
- Sunstone (the web-based GUI);
- MySQL DB backend (can be separate);
- API services (OCCI or EC2);
- advanced services (OneFlow, OneGate, etc.).

A control node unavailability does not affect running VMs, since it has only control on them (start & stop, monitoring, etc.).

An ONE infrastructure is remarkably flexible, and allow for two different strategies:

(i) to start from a very basic OS image, using tools to configure it and hence effectively decoupling OS from application. Such a strategy requires to maintain only a limited number of images, quite small being basic OS images, and the VMs can be adapted, once instantiated, to a specific scenario by contextualization scripts. Such scripts can be complex and difficult to maintain, and a complex contextualizations can be (very) slow.

(ii) to instantiate an OS image, configure it to the required level of complexity, save it and re-use it for subsequent instantiations. Although allowing for a short VM start-up time and providing always consistent saved images, such an approach leads to a large number of images to maintain; moreover fully configured images can be (very) large.

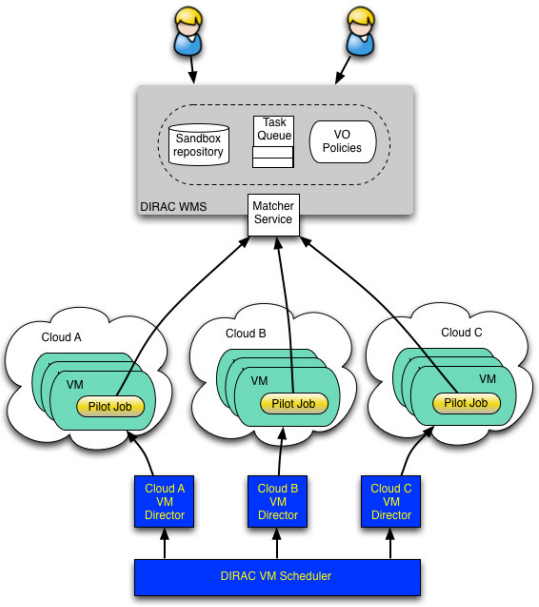

Figure 3. The VMDIRAC Virtual Machines management.

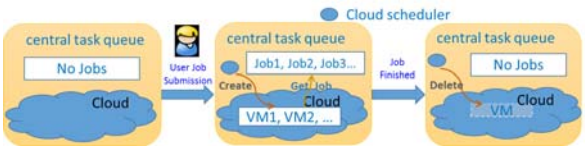

Figure 4. The elasticity approach within VMDIRAC.

3. VMDIRAC

The DIRAC [6] (Distributed Infrastructure with Remote Agent Control) system is a framework allowing for complex distributed environments, used by large HEP communities such as BESIII and BELLEII. Cloud resources can be connected to the DIRAC by the VMDIRAC [7] module, an extension of the DIRAC system that can dynamically and automatically activate, in case of an increasing demand for computing resources, new VMs with reserved job slots for new tasks. It can manage a pool of different kind of clouds by API drivers like EC2, OCCI, rOCCI, etc.

As the DIRAC pilot director can start pilot jobs, the VMDIRAC VM Director can instantiate pilot VMs, special VMs that at boot time start a pilot job and that behave just as other worker nodes w.r.t. the DIRAC Workload Management System (WMS). A "VM Scheduler" can dynamically manage VM according to status of the jobs and their queues.

VMDIRAC allows hence to integrate into DIRAC federated clouds as OCCI compliant CIs (OpenStack, ONE), CloudStack or Amazon EC2. This module can: check the Task Queue (TQ) and start VMs; contextualize such VMs in order to make them proper Worker Nodes (WNs) for the DIRAC WMS; pull jobs from central TQ; centrally monitor the status of the VMs and trigger an automatic shutdown according to jobs queues status.

The VMDIRAC architecture (Fig. 3) is composed of:

(i) on the DIRAC server side:
  (a) VM Scheduler: gets job status from TQ and matches it with the proper cloud site; submits requests of VMs to the Director;
  (b) VM Manager: retrieves statistics of VMs' status and establishes if new VMs are needed;
  (c) VM Director: connects with cloud manager to start new VMs;
  (d) Image Context Manager: contextualizes VMs to the specific required scenario.

(ii) on the VM side:
  (a) VM monitor Agent: periodically monitors the VMs' status and shutdowns VMs when not needed anymore;
  (b) Job Agent: acts just like pilot jobs, pulling jobs from TQ.

(iii) configuration and load management:
  (a) used to configure a CI joining the pool, and the images;
  (b) starts VMs;
  (c) run jobs on VMs.

VMDIRAC is hence an effective tool to centrally and remotely manage a (large) federation of (different) CIs, with a dynamic allocation and deallocation of computing resources (Fig. 4).

4. Automatic set-up of cloud (micro-)infrastructures

We developed and tested a tool, hosted on a small size bootable usb drive, to automatically deploy a CI of any size, from micro- and small infrastructures to medium size ones. Such usb drive automatically deploys an ONE cloud manager based on a CentOS 6.7 basic server OS (CentOS-6.7-x86_64-netinstall.iso). The process is automatic thanks to a Kickstart file defining all the parameters of the installation but the network parameters that are specific to every single infrastructure and therefore need to be manually input by the user. The same Kickstart file triggers the installation of all the required packages, and then proceeds to the contextualization of the infrastructure to the specific HEP scenario (like BESIII or BELLEII).

Template Kickstart files are also provided, downloadable from the Central Image Repository together with the needed customization shell script. By means of this script the user can easily

generate a customized Kickstart indicating the parameters that are specific and strictly related to the installation environment (e.g. root password, timezone, hostname, etc.).

The tools can be used on an empty host to deploy the infrastructure in around one hour, and then to instantiate and test VMs in order to validate the installation of the infrastructure. Additional hosts can be added to the infrastructure as well, by means of the very same usb drive, in order to increase the VMs capacity of the infrastructure.

VMs communicate with the Hypervisor and with each other through a private network (Fig. 5), hidden to the external public network. This solution provides multiple advantages: VMs are insulated from the public network so any network misconfiguration will not cause any damage to the public network where other servers may be connected; the Hypervisor itself acts as a firewall for the VMs, that are not exposed to the internet.

The deployment of the infrastructure proceeds first installing and setting up the cloud manager; once operational the cloud manager can exploit the VM Images and Templates available in a central Image Repository in order to make the infrastructure fully operational. An extensive testing stage is then performed to validate the deployed infrastructure.

4.1. Automatic set-up of the cloud manager

The installation of the cloud manager proceeds as follows:

- a virtualization environment is established, based on KVM.
- Ruby 1.8.7 is installed, with packages statically included in the tool to prevent effects of unexpected modifications of the Ruby packages in the public package repositories.
- OpenNebula 4.12 is installed, the *opennebula* and *opennebula-sunstone* services are started.
- configuration of the private network: a *br0* bridge is configured on the *em2* ethernet interface, the network configuration files are set accordingly, and the network is thus restarted; DHCP and DNS are then configured for the private network. The configuration is performed using a network template file, indicating the bridge and the private network parameters.
- a Squid proxy is installed. It acts like a cache that enables the storage of packages that would be frequently downloaded causing bottlenecks. The Squid Proxy is also used to speed up the access to CVMFS [8] repositories providing extra cache.
- IPTables rules are set, configuring the firewall in order to let VMs gain internet access, redirecting the traffic on the bridge *br0* to the ethernet interface *em1*. The rules needed to allow access to ports for the relevant installed packages are added as well.
- a rOCCI server, a Ruby implementation of OCCI, is installed on the Hypervisor, to provide an interface to perform common cloud operations through the Apache HTTP Server. Also in this case its installation exploits packages pre-cached on the usb drive, to decouple the tool from public package repositories. The certificate pem and key files are generated as well, and the ONE user *rocci* is created, in order to allow for the communication among the rOCCI server and the ONE backend. A rOCCI configuration file is created and added as module file for the Apache server to load. The server is then restarted. The rOCCI server allows to manage the infrastructure remotely by submitting jobs via VMDIRAC.

4.2. Automatic set-up of the VMs

The VMs can be instantiated in order to provide WNs fully customized to the tenants' needs. A central repository specific for each computing scenario (e.g. an HEP Collaboration as BESIII or BELLEII) provides VM Images and Templates that can be dynamically accessed by the cloud manager.

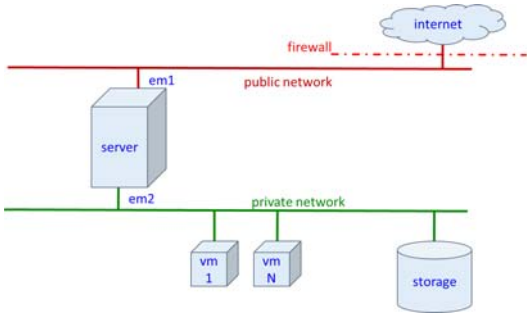

Figure 5. The network layout of the (micro-) cloud infrastructures installed by the tool.

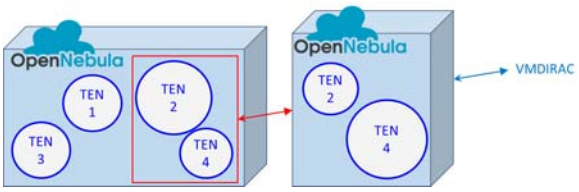

Figure 6. A modular approach of nested CIs: VMDIRAC feeds jobs and VMs to the "private" ONE CI (right), that cloud bursts to a second "public" CI (left).

Any OS can be used in principle within such an approach. For the time being we performed tests within the BESIII and BELLEII computing environments creating VMs with SLC6 as OS. To reduce the requirements on storage space, Collaboration software is downloaded each time a VM is instantiated; the Squid Server provides an extra cache for CVMFS. We chose to adopt QCOW2, an image format allowing for a dynamic increase of the storage space, reducing hence to the bare minimum the amount of data to be transferred, since all VMs point to the same disk image, and the write accesses to disk are directed to a separate, small and dedicated file.

Connectivity between ONE and the VMs is granted through the *one-context* and the *cloud-init* packages, that provide similar capabilities, since both allow to: set the public keys inside the VMs; set network parameters; configure some system parameters; mount the images; and run some scripts automatically. The *cloud-init* package offers moreover the possibility to use an higher level program for the contextualization stage.

Software repositories specific to the tenants' needs (e.g. the BESIII or the BELLEII simulation and reconstruction frameworks) can be mounted through CMVFS, eventually proxied by the Squid Server. There is no need to install the software in a VM, and only the required files are downloaded via HTTP to the VM. VM Templates that can be downloaded from the tenants' Image Repositories (proxied locally by the Squid Server) can then be used to instantiate VMs; a VM takes less than a minute to become fully operative.

While the creation of the VM Template requires a Virtual Manager and specific skills, such creation can be centrally performed by a group of maintainers for each tenant; the available VM Templates can then be exploited by all the sites of such tenants installing local (micro-)CIs.

4.3. Validation of the cloud (micro-)infrastructure

The whole set up of the newly installed CI, both the cloud manager and the VMs, can be validated by performing few tests.

The hypervisor must be accessible with the login credentials input to the Kickstart file, the network interfaces must be properly configured, and in case an external NAS is used as storage, it must be mounted in the proper location. Required services (as *opennebula*, *opennebula-sunstone* and *httpd*) must be running, and the *oneadmin* user must be able to log in as *root* on the Sunstone interface through the browser. The command *curl* can probe if the rOCCI server is listening on port 11443. Such tests can be automatically performed after the installation of the cloud manager by our tool, and a log file is stored for an inspection in case of problems.

The VMs must then be instantiated, in order to check that they can be deployed, access the software of the tenants and run jobs. The VM Image and Template are hence downloaded from the central Image Repository, and a new VM is instantiated. After its boot it is possible to

log in and verify that the contextualization has been correctly performed (e.g. running the test scripts that any modern HEP software suite provides to probe its installation). Also these tests can be automatically performed after the instantiation of the test VM by our tool, and a log file is stored for an inspection in case of problems.

After this first test of Sunstone, a further test is performed creating VMs and running jobs locally, and finally submitting jobs to the new infrastructure via VMDIRAC.

4.4. Image Repository

A central Image Repository can provide access to all the Images, Templates and scripts, to all the tenants (e.g. HEP Collaborations). Users can select the software version that is going to be used in the contextualization, eventually on VMs differing for OS or just for (some) libraries. The repository provides for the cloud manager server installation the Image, the Template and the Customization Script; all these are needed for the automatic installation. Template Kickstart files can be downloaded and customized by means of the customization script. The remote tenants can then dump the image on an usb drive for the automatic installation.

The repository provides also the VM images, that must be uploaded to the ONE Datastore via the Datastore file, and the Templates files, that must be imported as VM Templates. A VM Template will be instantiated each time new jobs need more computing resources.

5. Modular elasticity

The cloud (micro-)infrastructure deployed by our tool intrinsically allows for an on-demand usage, an elastic approach to cloud usage that does not occupy resources before jobs are submitted, and releases resources when not used, thus saving money both on academic and commercial clouds. VMDIRAC is one of the possible approaches allowing to use clouds elastically (other being for example HTCondor + Cloud scheduler [9], elastiq [10], etc.); it requires a central Task Queue and a cloud scheduler.

A deeper level of elasticity can be obtained even in a more modular approach (Fig. 6), exploiting the easy automatic set-up of (micro-)CIs and making use of two ONE instances: the remote VMDIRAC drives the private ONE, which cloud bursts to a second public CI.

The two ONE infrastructures show remarkably different scenarios:

- in the private ONE the stake-holder has full administrative control, can interface multiple tenants (i.e. multiple remote VMDIRACs), groups all the stake-holder's tenants, introducing one more layer in elasticity and acting as a simple tenant on the public ONE;
- in the public ONE the stake-holder acts as simple tenant and needs only user-level control.

As an example, consider a group of physicists funded to add cores, for HEP Collaborations A and B, to a ONE "public" infrastructure providing services to several groups and/or institutions. The group acts for the "public" ONE as a single stake-holder, retaining on the "private" ONE the full dynamic control on the resources sharing among A and B.

6. Conclusions

Several real-life scenarios could profit of a modular (almost) automatic set-up for elastic multi-tenants (micro-)CIs, as for example:

- consortium of independent tenants: independent experiments (HEP) or applications (SME) that want to act as a single stake-holder in large(r) CIs;
- stake-holders proxying theirs different tenants: experiments (HEP) or departments (Large Enterprises) with common resources procuring that want to decouple internal accounting/access to resources from large CIs;

- small groups (HEP) or applications (SME) with limited resources and cloud skills: micro-CIs exploiting automatic set-up can interface with remote VMDIRACs or incoming cloud-bursting.

The public Cloud Infrastructure (ONE or OpenStack) is not aware of the specific interaction with remote VMDIRACs (limited to the private ONs), the only interaction being a cloud bursting with private ONEs. The public CI gives stake-holders only user access to itself, and can still perform elastically with the different stake-holders. If the public CI is managed directly by the stake-holder, the latter can exploit automatic set-up of cloud (micro-)infrastructures and can receive cloud-bursting from trusted remote ONEs.

In the private ON CI the stake-holder has full control at administrative level. The ON CI interfaces with (remote) VMDIRACs for the different tenants of the stake-holder, that has full control of the tenants quotas in the private ON and hence on the public CI. It can perform elastically with the different tenants at the private ON level, and can elastically release resources on the public CI when not needed.

Even if the more complex modular approach were not be exploited, the availability of an easy and quick automatic set-up for elastic multi-tenants (micro-)CIs could allow small groups within large scientific Collaborations to migrate to a cloud computing approach even with no specific pre-existent skills and quite limited manpower. They could make their computing resources part of the Distributed Computing network of their Collaboration, installing with our tool the (micro-)CI and then letting a central remote VMDIRAC instance, typically hosted in the Collaboration main laboratory, manage the VMs and the job submission. Fully exploiting the Image Repository, even the maintenance and the update of the VMs and of their templates would be centralized, with enormously increased effectiveness and significantly reduced manpower.

Acknowledgments

The research activity described above has been performed within the framework of the BESIIICGEM Project (645664), funded by the European Commission within the call H2020-MSCA-RISE-2014. This work is also supported by the National Natural Science Foundation of China (NSFC) under grant no. 11375221.

References

[1] Zhang X M, Yan T, Zhao X H, Ma Z T, Yan X F, Lin T, Deng Z Y, Li W D, Belov S, Pelevanyuk I, Zhemchugov A and Cai H 2015 *Journal of Physics: Conference Series* **664** 032036 URL http://stacks.iop.org/1742-6596/664/i=3/a=032036

[2] Baranov A V, Balashov N A, Kutovskiy N A and Semenov R N 2016 *Physics of Particles and Nuclei Letters* **13** 672–675 ISSN 1531-8567 URL http://dx.doi.org/10.1134/S1547477116050071

[3] Bagnasco S, Berzano D, Guarise A, Lusso S, Masera M and Vallero S 2015 *Journal of Physics: Conference Series* **608** 012016 URL http://stacks.iop.org/1742-6596/608/i=1/a=012016

[4] Grzymkowski R, Hara T and computing group B I 2015 *Journal of Physics: Conference Series* **664** 022021 URL http://stacks.iop.org/1742-6596/664/i=2/a=022021

[5] Montero R S, Moreno-Vozmediano R and Llorente I M 2012 *Computer* **45** 65–72 ISSN 0018-9162

[6] Casajus A, Graciani R, Paterson S, Tsaregorodtsev A and the Lhcb Dirac Team 2010 *Journal of Physics: Conference Series* **219** 062049 URL http://stacks.iop.org/1742-6596/219/i=6/a=062049

[7] Albor V F, Miguelez M S, Pena T F, Muoz V M, Silva J J S and Diaz R G 2014 *Journal of Physics: Conference Series* **513** 032031 URL http://stacks.iop.org/1742-6596/513/i=3/a=032031

[8] Blomer J, Buncic P, Charalampidis I, Harutyunyan A, Larsen D, and Meusel R 2012 *Journal of Physics: Conference Series* **396** 052013 URL http://stacks.iop.org/1742-6596/396/i=5/a=052013

[9] Panitkin S *et al.* 2014 *Journal of Physics: Conference Series* **513** 062037 URL http://stacks.iop.org/1742-6596/513/i=6/a=062037

[10] Berzano D, Blomer J, Buncic P, Charalampidis I, Ganis G, Lestaris G and Meusel R 2014 *Journal of Physics: Conference Series* **513** 032007 URL http://stacks.iop.org/1742-6596/513/i=3/a=032007