Adding Virtualization Capabilities to the Grid'5000 Testbed*

Daniel Balouek[1], Alexandra Carpen Amarie[1], Ghislain Charrier[1], Frédéric Desprez[1], Emmanuel Jeannot[1], Emmanuel Jeanvoine[1], Adrien Lèbre[2], David Margery[1], Nicolas Niclausse[1], Lucas Nussbaum[3], Olivier Richard[4], Christian Perez[1], Flavien Quesnel[2], Cyril Rohr[1], and Luc Sarzyniec[3]

[1]INRIA, France
[2]Ecole des Mines de Nantes, France
[3]Université de Lorraine, France
[4]Université de Grenoble, France
FirstName.LastName@inria.fr, FirstName.LastName@mines-nantes.fr, FirstName.LastName@univ-lorraine.fr, FirstName.LastName@imag.fr

Abstract. Almost ten years after its premises, the Grid'5000 testbed has become one of the most complete testbed for designing or evaluating large-scale distributed systems. Initially dedicated to the study of High Performance Computing, the infrastructure has evolved to address wider concerns related to Desktop Computing, the Internet of Services and more recently the Cloud Computing paradigm. This paper present recent improvements of the Grid'5000 software and services stack to support large-scale experiments using virtualization technologies as building blocks. Such contributions include the deployment of customized software environments, the reservation of dedicated network domain and the possibility to isolate them from the others, and the automation of experiments with a REST API. We illustrate the interest of these contributions by describing three different use-cases of large-scale experiments on the Grid'5000 testbed. The first one leverages virtual machines to conduct larger experiments spread over 4000 peers. The second one describes the deployment of 10000 KVM instances over 4 Grid'5000 sites. Finally, the last use case introduces a *one-click* deployment tool to easily deploy major IaaS solutions. The conclusion highlights some important challenges of Grid'5000 related to the use of OpenFlow and to the management of applications dealing with tremendous amount of data.

Keywords: Distributed Systems, Large-Scale Testbed, Virtualization, Cloud Computing, Experiments.

1 Introduction

The evolution of technology allows larger and highly distributed systems to be built, which provide new capabilities, in terms of applications as well as in terms of

* The Grid'5000 experimental testbed and all development actions are supervised and financed by the INRIA ALADDIN framework with support from CNRS, RENATER, and several Universities as well as other funding bodies (see https://www.grid5000.fr). Grid'5000 experiments are partially supported by the INRIA large scale initiative Hemera. The IaaS deployment utility is a particular action developed with the support of the EIT ICT Labs.

I. Ivanov et al. (Eds.): CLOSER 2012, CCIS 367, pp. 3–20, 2013.

infrastructures like peer-to-peer systems, Grids, and more recently (federations of) Cloud platforms. Such large scale distributed and parallel systems raise specific research issues and computer science, as other sciences, needs instruments to validate theoretical research results as well as software developments. Although simulation and emulation are generally used to get a glance of the behavior of new algorithms, they use over-simplified models in order to reduce their execution time and thus cannot be accurate enough. Leveraging a scientific instrument to perform actual experiments is a undeniable advantage. However conducting experiments on real environments is still too often a challenge for researchers, students, and practitioners: first, because of the unavailability of dedicated resources but second also because of the inability to create controlled experimental conditions, and to deal with the so large variability of software requirements. Started in 2003 under the initiative of the French ministry of Research, the Grid'5000 testbed is a scientific instrument for the study of large scale parallel and distributed systems. With the aim of providing a highly reconfigurable, controllable and monitorable experimental platform [14], Grid'5000 was solid enough to attract more than 600 users and led to a large number of research results and publications. Nowadays, Grid'5000 is internationally recognized and serves as a foundation for new scale platforms, e.g. FutureGrid [17] in the USA. With almost ten years of background, several members of its scientific or technical board are invited take part to different working groups, events focusing on the design and the building of new experimental testbeds [16,27] with the ultimate objective of improving the quality of experiments.

The Grid'5000 instrument is continuously evolving toward providing more flexibility, more control of both the electronic devices composing the infrastructure as well as of the experiments running over. The scientific and technical boards carefully follow the major trends and the latest innovations of distributed and parallel systems from both hardware and software point of views. This enables to renew the infrastructure while ensuring the delivering of a testbed that meets user-expectations. As an example, one of the most important change of the last decade is the renewal of interest of virtualization technologies. The virtual machine concept that enables to run any system over any other one has radically changed the use of distributed systems, leading to new large-scale platforms built upon shared data-centres and usually classified into the new cloud-computing IaaS (Infrastructure-as-a-Service) paradigm. Indeed, in addition to abstract the complexity of IT systems, the use of virtualization is motivated by the fact that physical resources are usually under-used and that virtualization technologies enable to consolidate them and thus improve the productivity of the whole platforms.

Considering that the current trend consists of "virtualizing" all physical resources, adding virtualization capabilities to Grid'5000 is obviously expected. From the end-users point of view, the objective is twofold: first, it will enable to leverage virtualization technologies to improve the quality of the experiments at a larger scale. Second, it will enable to investigate new concerns related to the management of virtualized infrastructures. Indeed, despite of the tremendous progress in the virtualization area and the large number of companies providing virtualized platforms for various users, several important issues remain to be solved. Among them, Quality of Service (QoS), fault-tolerance, energy management, and scalability are major ones. Extending the Grid'5000 software and services stack to investigate such concerns is important for the community. The key

progress, beyond the state of the art, is to provide the user with an infrastructure where each component can be virtualized. In addition to the system virtualization capabilities provided by modern computers, Grid'5000 targets the virtualization of active network equipments as well as storage facilities.

In this paper, we describe the latest contributions of the Grid'5000 software and services stack to make large-scale experiments involving low level virtual technologies up to full IaaS software stacks. Grid'5000 is one the very few platforms that allows to conduct such experiments between multi-sites and in an isolated and reproductible manner.

The reminder of this paper is structured as follows. In Section 2, we give an overview of the Grid'5000 instrument. Section 3 describes the latest contributions of the Grid'5000 software and service stack while Section 4 illustrates the use of such contributions through discussing three use-cases. Other experimental testbeds are introduced in Section 5. Finally, we discuss perspectives and conclude this article in Section 6.

2 Grid'5000 Overview

In 2003, several teams working around parallel and distributed systems designed a platform to support experiment-driven research in parallel and distributed systems. This platform, called Grid'5000 [14] and opened to users since 2005, was solid enough to attract a large number of users. It has led to a large number of research results: 575 users per year, more than 700 research papers, 600 different experiments, 24 ANR projects and 10 European projects, 50 PhD, and the creation of startup companies as well.

Grid'5000 is located mainly in France (see Figure 1), with one operational site in Luxembourg and a second site, not implementing the complete stack, in Porto Alegre, Brazil. Grid'5000 provides a testbed supporting experiments on various types of distributed systems (high-performance computing, grids, peer-to-peer systems, cloud computing, and others), on all layers of the software stack. The core testbed currently comprises 10 sites. Grid'5000 is composed of 26 clusters, 1,700 nodes, and 7,400 CPU cores, with various generations of technology (Intel (60%), AMD (40%), CPUs from one to 12 cores, Myrinet, Infiniband {S, D, Q}DR and 2 GPU clusters). A dedicated 10 Gbps backbone network is provided by RENATER (the French National Research and Education Network). In order to prevent Grid'5000 machines from being the source of a distributed denial of service, connections from Grid'5000 to the Internet are strictly limited to a list of whitelisted data and software sources, updated on demand.

From the user point of view, Grid'5000 is a set of sites with the exact same software environment. The driving idea is that users willing to face software heterogeneity should add controlled heterogeneity themselves during their experiments. Three basic workflows are supported when staging an experiment on Grid'5000: a web interface-based workflow, an API-based workflow, and a shell-based workflow. These differ not only in the interfaces used but also in the process they support.

The core steps identified to run an experiment are (1) finding and booking suitable resources for the experiment and (2) deploying the experiment apparatus on the resources. Finding suitable resources can be approached in two ways: either users browse a description of the available resources and then make a booking, or they describe their

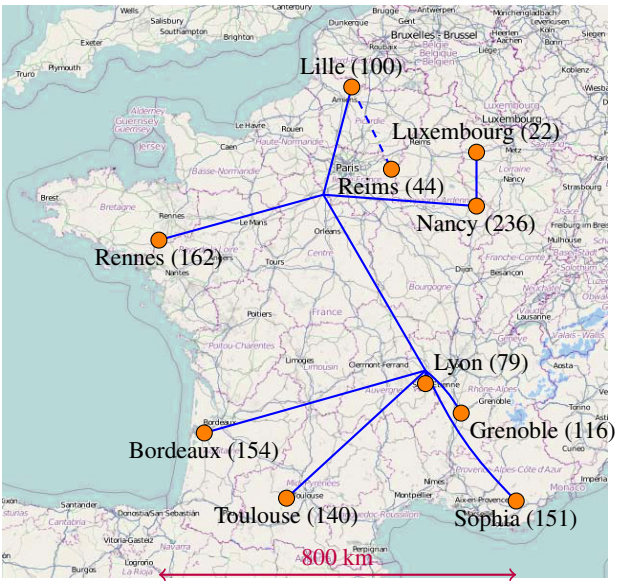

Fig. 1. Grid'5000 sites and their number of nodes

needs to the system that will locate appropriate resources. We believe both approaches should be supported, and therefore a machine-readable description of Grid'5000 is available through the reference API. It can be browsed by using a web interface or by running a program over the API. At the same time, the resource scheduler on each site is fed with the resource properties so that a user can ask for resources describing the required properties (e.g., 25 nodes connected to the same switch with at least 8 cores and 32 GB of memory). Once matching resources are found, they can be reserved either for exclusive access at a given time or for exclusive access when they become available. In the latter case, a script is given at reservation time, as in classical batch scheduling.

Several tools are provided to facilitate experiments. Most of them were originally developed specifically for Grid'5000. Grid'5000 users select and reserve resources with the OAR batch scheduler [13,30]. Users can install their own system image on the nodes (without any virtualization layer) using Kadeploy [18]. Experiments requiring network isolation can use KaVLAN to reconfigure switches and isolate nodes from the test of the testbed. Several monitoring tools (resource usage on nodes with Ganglia, energy consumption) are also available. All tools can be accessed by a REST API to ease the automation of experiments using scripts. The tools used to support the experiments over Grid'5000 will be described in Section 3.

Different approaches to deploying the experimental apparatus are also supported. At the infrastructure level users either use the preconfigured environment on nodes, called the production environment, or they install their own environment. An environment consists of a disk image to be copied on the node and of the path in the disk image of the kernel to boot. This environment can be prepared in advance by modifying and saving reference environments made available to users, or a reference environment can be dynamically customized after it is deployed on the resources. The approach chosen

can affect the repeatability of the results. Therefore, choices concerning the experiment testbed environment are left to the experimenters.

Whatever approach used for the first two steps described here, access to resources (sites and nodes) is done through SSH. Each site has its own NFS server. This design decision was taken to ensure that resources of a particular site can be used even when the link to other sites is undergoing maintenance. In other words, the infrastructure does not depend on a single site to stay operational—an important consideration because maintenance events become frequent when 10 sites are operated.

3 A Software Stack to Support Experiments

This section describes four key Grid'5000 services that contribute to support virtualization and Cloud experiments on Grid'5000. *Kadeploy* (Section 3.1) enables users to deploy their software stacks of choice on the nodes. *g5k-subnets* (Section 3.2) and *KaVLAN* (Section 3.3) provide two different ways to configure the network (respectively by reserving IP address ranges, and by isolating an experiment from the rest of the testbed using on-the-fly switches reconfiguration). Finally, the Grid'5000 REST API (Section 3.4) uniformizes the access to those services that facilitate the automated execution of experiments.

3.1 Providing Custom Experimental Environments with *Kadeploy*

On most clusters, users do not have the option of changing the operating system installed on nodes. This is a severe problem for experimentation, since experimenters often need to perform experiments in many different contexts in order to extend the scope of an experimental result by verifying that it is not limited to specific experimental conditions (specific kernel, library or compiler version, configuration, etc.).

Grid'5000 enables the deployment of custom software stacks (including the operating system) on bare hardware[1]. This allows users to perform experiments without being bound to one particular Linux distribution or version, or even operating system. Users could use their own modified Linux kernels to work on live migration or memory deduplication techniques, or even install FreeBSD or Solaris to evaluate the interest of process containers available on those operating systems (such as FreeBSD Jails or OpenSolaris Zones) for Cloud computing.

While it is common for Cloud infrastructures to provide the ability to deploy custom OS images in virtual machines, Grid'5000 provides this feature on physical machines, which brings two advantages. First, it avoids the overhead of the virtualization layer, which can be a problem when doing experiments involving performance measurements. While the overhead is extremely low for CPU-intensive workload, it can be much higher for IO-intensive workloads. Second, it allows deployed environments to contain virtual machines themselves, without requiring the use of *nested* virtualization (hypervisor inside a virtual machine), which is not supported very well by today's hypervisors.

On Grid'5000, the installation of custom OS images on nodes is implemented using

[1] This has been recently named as Hardware-as-a-Service.

the *Kadeploy* [18] cluster provisioning system, which has been developed in the context of the Grid'5000 project. *Kadeploy* achieves efficient and scalable installation of system images using advanced mechanisms (adaptative tree-based command execution thanks to TakTuk [15]; chain-based image broadcast [18]). The deployment process is controlled by an automata to handle the unavoidable errors (due to unreliable protocols and hardware), and the corresponding retry policies. Thanks to those features, the installation of a 1.5 GB image on 130 nodes takes less than 10 minutes. Additionally, instead of restricting deployments to the system administrator, *Kadeploy* provides flexible permissions management to allow users to start deployments on their own. This is used on Grid'5000 to enable users to deploy their own deployment environments.

Grid'5000 users can provide their own deployment images, and install them on nodes with no prior validation from the technical team. While minor problems have been encountered (e.g. a FreeBSD network driver that was disabling – until the next reboot – the IPMI implementation sharing the Ethernet port with the operating system), no major problem has been encountered due to this policy. This is also an example of the security policy that is deployed throughout Grid'5000. We focus on mitigating normal user errors, and on checking users before giving them access to the testbed, but we do not try much to fight malicious actions from users since this would often limit the experimental capabilities of the testbed at an unacceptable level.

3.2 Network Reservation with *g5k-subnets*

Virtual machines used during experiments must be accommodated on the testbed's network. While it is sometimes possible to limit experiments to purely virtual networks (inside one physical machine, or spanning several physical machines using e.g. Open vSwitch), this would be a severe limitation. Additionally, Grid'5000 is composed of several sites with routing between sites (Figure 1), and different users can run concurrent experiments on the same Grid'5000 site.

Therefore, techniques to reserve address ranges or to isolate an experiment from the rest of the testbed are needed. Grid'5000 provides two such solutions: *g5k-subnets* (described in this section) extends Grid'5000 resource reservation mechanism to allow users to reserve IP ranges for their virtual machines; *KaVLAN* (presented in the next section) reconfigures network switches so that an experiment is isolated from the rest of the testbed.

The whole $10/8$ subnet ($10.0.0.0 - 10.255.255.255$) is dedicated to user virtual machines on Grid'5000. The first half ($10.0 - 10.127$) is used for KaVLAN, while the second half ($10.128 - 10.255$) is used by *g5k-subnets*. Since Grid'5000 sites are interconnected via L3 routing, the $10.128/9$ network is divided into one $/14$ network per site ($2^{18} = 262144$ IP addresses per site). This $/14$ network per site is again divided, with the last $/16$ network ($2^{16} = 65536$ IP addresses) dedicated to attributing IP addresses over DHCP for machines in the `00:16:3E:XX:XX:XX` MAC range (which is the Xen reserved MAC range).

The last $3 * 2^{16} = 196608$ IP addresses are allocated through reservation with *g5k-subnets*. *g5k-subnets* is integrated in the *Resource Management System* used on Grid'5000, OAR [30]. Users can reserve a set of network IP addresses (from $/22$ to a

/16) at the same time as nodes: the following command reserves two /22 ranges and 8 nodes:

```
oarsub -l slash_22=2+nodes=8 -I
```

Once a specific IP range has been allocated, users can retrieve it using a command-line tool. Additional information, such as DNS servers, default gateway, broadcast address, etc. is made available through this tool.

It is worth noting that *g5k-subnets* only manages the reservation of IP address ranges, not of MAC addresses. Since the available MAC address range (47 bits, since one is used to indicate multicast frames) is much larger than the available IP range (18 bits), choosing MAC addresses at random does not result in significant chances of collision. This strategy is also used by several Cloud software stacks.

Finally, *g5k-subnets* does not enforce the reservation. A malicious user could *steal* IP addresses from a concurrent user. If a user requires stronger protection, the use of KaVLAN is recommended.

3.3 Network Isolation with *KaVLAN*

In some cases, the reservation of IP ranges, as provided by *g5k-subnets*, is not sufficient to satisfy the experimenters' needs. Some experiments are either too sensitive to external noise (coming from broadcasts, or from unsolicited connections), or too disruptive (e.g. when using network discovery protocols that rely on network broadcast). A typical example in experiments involving virtualization is the installation of a DHCP server to serve IP addresses to virtual machines. If not properly configured, it could start answering DHCP requests from other nodes on the testbed. Such experiments cannot be performed on the same network as other experiments, as they could compromise the testbed's infrastructure or other experiments, or be compromised themselves.

KaVLAN is a tool developed inside the Grid'5000 project that provides controlled isolation of user experiments at the network level. *KaVLAN* isolates experiments in their own 801.1q VLAN by reconfiguring the testbed's switches for the duration of the experiment. It can connect to switches using SNMP, SSH and telnet, supports a number of different routers and switches (from Cisco, HP, 3com, Extreme Networks and Brocade), and can easily be extended to support other products.

Several different types of VLANs are provided by *KaVLAN* to meet different user needs (Figure 2):

- **Local VLAN** provides users with a fully isolated network that is only accessible by connecting (generally using SSH) from a machine connected to both the VLAN and the testbed's network;
- **Routed VLAN** also provides users with a separate L2 network, but that network can be reached from any node of the testbed since the network is routed by the site's router. It can typically be used to deploy a complex infrastructure including a DHCP server (e.g. a Cloud middleware) inside the VLAN.
- Instead of providing isolation limited to one site (as with local and routed VLAN), a **Global VLAN** provides a separate L2 network at the scale of the testbed, using 802.1ad (Q-in-Q) on the testbed's backbone network. It is accessible from the default testbed's network using routing.

VLAN type	Ethernet isolation	IP isolation	Multi-site	# of VLAN
local	yes	no	no	3 per site
routed	yes	no	no	3+3 per site
global	yes	no	yes	1 per site

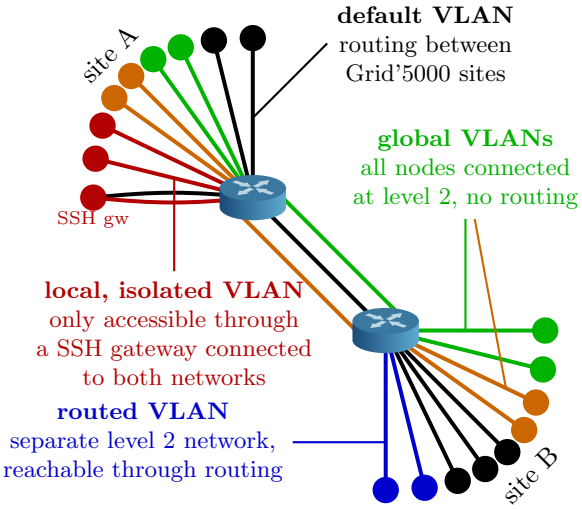

Fig. 2. Types of VLAN provided by *KaVLAN*

KaVLAN is also used on Grid'5000 in order to provide temporary interconnections with other testbeds. For example, nodes can be removed from Grid'5000, and integrated in another testbed, for the duration of an experiment.

3.4 Providing a Unified Interface with a REST API

Some Grid'5000 services are traditionally used through command-line interfaces. While this a good step towards enabling the automation of experiments through scripting, it still has a few limitations:

- Developing user-friendly command-line interfaces is hard and time-consuming.
- Ensuring consistency between several tools on the naming of parameters or the formatting of outputs is hard, and even harder if backward compatibility must be supported.
- Several tools output large volumes of structured data. In that case, parsing the output of a command in a script is inconvenient, as there is often a need to handle error conditions at the same time.
- Running external commands from scripts is inconvenient, since those commands often need to be executed on specific machines over SSH.

In order to overcome those limitations in Grid'5000, the focus has been put in providing a consistent REST API that provides access to the various Grid'5000 services. The Grid'5000 API is composed of several more focused APIs:

Reference API. This API gives access to a detailed description of most elements of the testbed, such as nodes (with their hardware description) and network equipments and links. This API can be used by users to find resources with specific characteristics (e.g. node with Intel Nehalem architecture, and at least 24 GB or RAM), or to ensure that nodes are still conforming to their description – a tool implementing this verification runs on nodes at each boot.

Monitoring API. This API provides the state of node (available for reservation, used by a job currently running on the testbed, etc.). It can be used by users, in combination with the Reference API, to find available resources matching their needs.

Metrology API. This API provides a common interface to various sensors, either software (e.g. Ganglia) or hardware (e.g. energy consumption). Custom metrics can also be added. It is aimed at providing users with the performance status of their nodes during their experiments.

Jobs API. While the OAR resource management system is traditionally used through a command-line interface, this API provides a REST interface to submit and manage jobs.

Deployments API. Similarly to the Jobs API, the Deployments API provides a higher-level interface to Kadeploy.

Several interfaces have been developed on top of the Grid'5000 API. First, a web interface enables users to perform most actions, including resource selection (using the Reference API) and reservation (using the Jobs API). Command-line tools have also been developed. For example, *g5k-campaign* aims at orchestrating experiments startup. It is featured in Section 4.3 where it is used–with custom engines–to deploy Cloud frameworks.

4 Grid'5000 and Virtualization Capabilities: Use-cases

This section presents three use-cases that leverage latest contributions and system virtualization as building blocks. In the first one, virtualization is used as a mean to temporary emulate a larger testbed composed of 4000 peers. In the second one, a set of scripts that enables the deployment of a significant number of VMs upon Grid'5000 is presented. Thanks to these scripts, end-users may investigate particular concerns related to the management of large-scale virtualized infrastructures at low-level. The last one deals with the automation of IaaS deployment. Lot of Grid'5000 users want to investigate the impact of the virtualization layer on a particular workload. Delivering a tool that relieves end-users with the burden of deploying and configuring an IaaS system is a real advantage. In such scenarios, Grid'5000 is seen as an IaaS platform where end-users may provision VMs according to the needs of the applications. Although adding virtualization capabilities to Grid'5000 is an on-going task targeting the virtualization of all devices, we believe that these three use-cases are already representative of a wide scope of experiments.

4.1 Testing the Scalability of Kadeploy by Deploying 4000 Virtual Machines

Large-scale testbeds are a rare resource. With its 1300+ nodes, Grid'5000 is already one of the largest experimental testbeds. However, its size can still be a limiting factor for

some experiments. One example of such experiments is the evaluation of the suitability of *Kadeploy* (presented in Section 3.1) to manage Exascale clusters, which can be composed of thousands of compute nodes. On Grid'5000, Kadeploy is installed using one separate installation per site, rather than one global installation, which does not reflect the configuration expected on Exascale clusters, with only one installation managing all the nodes.

We therefore performed a set of experiments on Grid'5000 to evaluate the performance of Kadeploy when used to manage a 4000-nodes cluster [26]. In order to create a level-2 network to accomodate all the virtual machines, we used a global KaVLAN network spanning four sites with a diameter of 1000 km. 668 nodes where used during that experiment (out of 783 available with the required capabilities). 635 were used to accomodate 3999 KVM virtual machines (managed using custom-made scripts), while the remaining 33 nodes where used to host the Kadeploy server, a DNS server, a DHCP server, and HTTP servers used to serve the minimal system image used during the Kadeploy deployment.

The automated configuration of our 4000-nodes Kadeploy testbed took 40 minutes, decomposed in: 20 minutes to reserve and deploy 668 Grid'5000 nodes; 5 minutes to prepare all physical nodes; 15 minutes to instantiate the 4000 virtual machines. At this point, it was possible to perform Kadeploy deployments over all the virtual machines. We performed a successful deployment of 3838 virtual machines using a 430 MB-environment in 57 minutes.

While the success of this experiment demonstrates the ability of Kadeploy to manage clusters of 4000 nodes as well as the adequacy of Grid'5000 to perform large-scale experiments in virtualized environments, it also allowed us to identify some bottlenecks in Kadeploy, which opened the path for future works.

4.2 Playing with VMs at Large-Scale

Live-migration of virtual machines is one of the key-point of virtualization technologies. Besides simplifying maintenance operations, it provides an undeniable advantage to implement fine-grained scheduling policies such as consolidation or load-balancing strategies.

However, manipulating VMs throughout a large-scale and highly-distributed infrastructure as easy as traditional OSes handle processes on local nodes is still facing several issues. Among the major ones, we can notice the implementation of suited mechanisms to efficiently schedule VMs and to ensure the access to the VM images through different locations. Such mechanisms should assume to be able to control, monitor, and communicate with both the host OSes and the guest instances spread across the infrastructure at any time. If several works have addressed these concerns, the real experiments are in most cases limited to few nodes and there is a clear need to study such concerns at higher scales. With this objective in mind, a set of scripts[12] have been designed over the Grid'5000 software stack. They allow us to easily start a significant number of KVM instances upon several sites of the testbed. These instances can then be used at user convenience in order to investigate particular concerns such as, for instance, the impact of migrating a large amount of VMs simultaneously or the study of new proposals dealing with VM images management. Through the use of a global VLAN (Section 3.3), the

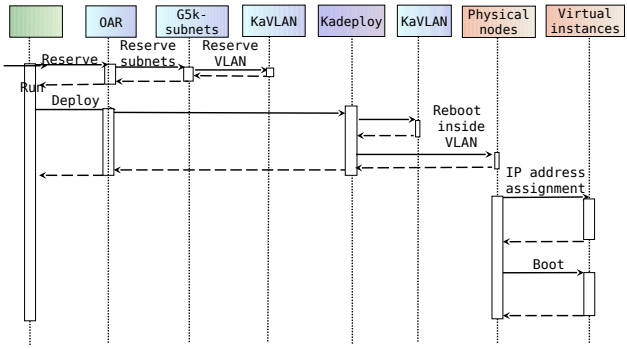

Fig. 3. Sequence diagram of the infrastructure installation

user may choose to virtualize all sites as a unique one or not. This enables to avoid network domain issues when a VM is migrated from one network to another one.

To deliver such a setup, the script goes through 3 logical steps:

Booking Resources. Using the *disco* tool that provides multi-criteria and multi-site search for available Grid'5000 resources, the first script is in charge of finding the available nodes that support hardware virtualization, booking them and requesting network resources (i.e. a /18 subnet for the IPs and a global VLAN if need be). These resources are mandatory to deal with IP assignment and routing within the infrastructure.

Deploying and Configuring Physical Machines. This task consists of deploying bare-metal hypervisors and installing the packages related to the virtualization on the host machines. It is worth noting that during the deployment phase, an additional option of *Kadeploy* enables to reboot each physical machine inside a particular VLAN. The script is leveraging this argument if the experiment involves several sites and a global VLAN has been booked. At the end of the deployment, the global routing is configured on each node and the network is isolated from the usual routing policy (cf Section. 3.3).

Starting the Virtual Machines. The virtual instances are started simultaneously, using a hierarchical structure among the physical nodes. Each virtual machine receives an IP address and a name leveraging *g5k-subnets* and a round robin assignment policy. The correlation between name and IP is stored in a dedicated file propagated on each physical node. This allows us to identify and communicate with all the virtual machines. Finally, the name and the IP of each VM are configured by customizing the related copy-on-write image before booting it.

The sequence diagram in Figure 3 illustrates these different steps.

Deploying such a large number of VM instances led to several concerns and the use of additional scripts has been required. Leveraging Taktuk [15], these scripts are used to propagate virtual machines images on each bare metal, to communicate with all the virtual instances to check whether the VMs are up or not and to control the state of the whole system during the execution of experiments.

```
deployment:
  engine:
    name: opennebula
  sites:
    rennes:
      nodes: 5
      subnet: slash_22=1
  walltime: 2:00:00
opennebula:
  controller_user: "oneadmin"
  controller_group: "cloud"
  hypervisor: kvm
  datastore:
    ONstore:
      filesystem: hdfs
  vmimage:
    ttylinux:
      path: /tmp/openNebulaImages/ttylinux.img
      datastore: "ONstore"
```

Fig. 4. Configuration file for the OpenNebula g5k-campaign engine

Considering that physical machines must support hardware virtualization to start KVM instances, the largest experiment that has been conducted up to now involved 10240 KVM instances upon 512 nodes through 4 sites and 10 clusters. The whole setup is performed in less than 30 minutes with about 10 minutes spent on the deployment of the nodes, 5 minutes for the installation and configuration of the required packages on the physical hosts, while 15 minutes are dedicated to the booting of the virtual machines. The result of that work opens doors to the manipulation of virtual machines over a distributed infrastructure like traditional operating systems handle processes on a local node. This new functionality is currently used to validate large scale algorithms in charge of managing virtualized infrastructures such as [24].

4.3 Delivering Cloud Platforms in *One-Click*

Although Cloud Computing is gaining consensus from both scientific and industrial communities, its usage still faces some concerns that limit its adoption. The impact of the virtualization technologies, the reliability of virtualized environments and the lack of advanced provisioning technics are some examples of such concerns.

They are at the core of a new research direction targeted by the Grid'5000 community, aiming at enabling experimental research at all levels of the Cloud Computing stack. The first step towards investigating Infrastructure-as-a-Service concerns within Grid'5000 was achieved through a set of "sky computing" tools [25]. Such tools enabled large-scale experiments that spanned across Grid'5000 and FutureGrid [17], harnessing over 1500 cores for a federation of several Nimbus Clouds [19]. These experiments showed that testbeds such as Grid'5000 may play an essential role in providing researchers with configurable Cloud platforms similar to commercially available Clouds.

However, the complexity of managing the deployment and tuning of large-scale private Clouds emerged as a major drawback. Typically, users study specific Cloud components or carry out experiments involving applications running in Cloud environments. A key requirement in this context is seamless access to ready-to-use Cloud platforms,

as well as full control of the deployment settings. To address these needs, a *one-click* deployment tool for Infrastructure-as-a-Service environments has been developed [21].

***One-click* IaaS Clouds with g5k-Campaign.** The deployment utility is designed to install and configure fully-functional Cloud platforms over Grid'5000 in a fast and reliable manner. The current version of the system supports two open-source IaaS Clouds, namely OpenNebula [20,22] and Nimbus [19,29].

The deployment tool is built on top of *g5k-campaign*, a framework devised for coordinating experiment workflows and launching repeatable experiments on Grid'5000. *G5k-campaign* relies on extensible *engines* to describe experiments. Such engines define the stages of an experiment: physical node reservations in Grid'5000, environment deployment, configuration, and experiment execution.

To simplify user interaction with the Cloud deployment tools, the *g5k-campaign* framework has been enhanced with a simple, yet powerful mechanism to customize experiments. It relies on configuration files to specify user requirements in terms of reserved nodes and Cloud environment settings, which are then transparently configured during the execution of the deployment engine.

A configuration file example is provided in Figure 4. It consists of several YAML indented blocks that account for the various steps of the deployment process. The *deployment* block includes Grid'5000 node reservation details, such as the sites to be reserved and the number of nodes for each of them. The *opennebula* block comprises configuration options for OpenNebula, ranging from user information to VM storage mechanisms and APIs. Note that users can also describe virtual machine images in the *vmimage* sub-block, to automate image uploading into the OpenNebula system.

A wide range of Cloud-specific parameters can thus be managed by the deployment tools, including hypervisor and virtualization settings, host nodes configuration, installation of external packages, authentication settings, virtual networks creation, configuration of the various storage mechanisms for VM images and of the Cloud user interfaces.

The implementation of the Cloud deployment tools heavily relies on the latest version of the Grid'5000 software stack introduced in Section 3. First, to provide support for virtualization and full control over the environment, the Cloud platforms are installed on standard environments deployed on the physical machines through *Kadeploy*. The interaction with the Grid'5000 services is implemented on top of the *Grid'5000 API*, which is in charge of managing the node reservations and deployments, as well as of retrieving the available nodes and reporting errors. Another essential building block is represented by the *g5k-subnets* tool. It provides the virtual networks needed by the Cloud services to equip VMs with appropriate IP addresses on each site.

Zoom on the OpenNebula Deployment Engine. The engine is responsible for handling the installation process of the OpenNebula environment, either from Debian packages or from specific source code archives. It automatically carries out the deployment and configuration, with a particular focus on storage mechanisms for virtual machines. Currently, the OpenNebula engine supports ssh-based image propagation and shared storage based on NFS (for single-site deployments) or HDFS [28] (for multi-site deployments), to enable live migration and enhance scalability.

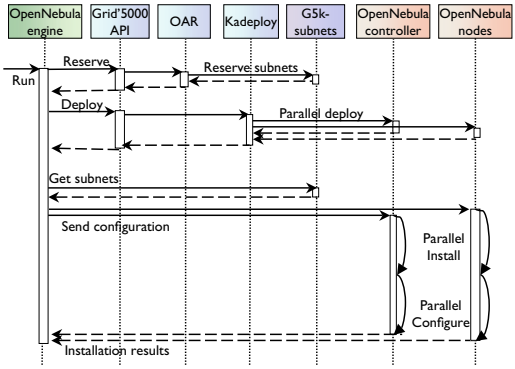

Fig. 5. Sequence diagram of an OpenNebula engine execution

The OpenNebula engine can be executed by passing a configuration file, such as the one given in Figure 4, to the *g5k-campaign* tool, which is in charge of interpreting it and delivering the ready-to-use Cloud platform, as in the following command:

```
g5k-campaign -C opennebulaMultisite.yml
```

The sequence diagram in Figure 5 describes the execution workflow of the OpenNebula engine. First a node reservation is made for each site specified in the configuration file through the *Grid'5000 API*. Along with the nodes, the *OAR* system also reserves a range of virtual IPs corresponding to each site. The next step is the parallel deployment of one or more environments on the reserved nodes enabled by *Kadeploy*. Once the nodes are operational, the OpenNebula engine retrieves the reserved IP ranges from each site and then creates specific configuration settings for each node, according to their role (e.g., the OpenNebula controller is assigned the list of host nodes). Finally, OpenNebula is installed and configured on each node in parallel and the outcome of these processes is returned to the engine. When the execution of the engine is successfully completed, the user can access and perform experiments on the deployed Cloud platform, for the duration of the Grid'5000 reservation defined in the configuration file. These execution stages apply to both multi-site and mono-site deployments, as their outcome is similar: a single Cloud comprising one controller and a set of host nodes. The specificity of a multi-site Cloud is that it will have access to several virtual networks, each of them corresponding to a group of host nodes belonging to the same site.

The OpenNebula deployment engine is written in Ruby and the installation and configuration are done on each physical node by using the Chef [23] configuration management framework. The Chef recipes are designed in a modular manner, to allow Cloud users to add or extend the current OpenNebula configuration options. This tool was validated by installing OpenNebula on 80 physical nodes belonging to 3 Grid'5000 sites, on which we deployed 350 virtual machines. The average time to deploy such a ready-to-use OpenNebula Cloud is less than 20 minutes, with about 6 minutes spent on infrastructure installation and configuration, while the rest is taken up by nodes reservation and deployment. Moreover, subsequent re-deployments take only 5 minutes, as the environments are already running and required packages are installed.

5 Related Work

Several experimental platforms exist over the world for different target sciences.

Around network and system research, Emulab [4] is a network testbed made available to the international academic community since 2001. The original motivation is to provide a single site where users can deploy and execute replayable networked experiments on dedicated hardware. The platform provides customizable network and servers but it is not designed nor sized to host numerous and large experiments related to virtualization, storage or power management. Protogeni [10] is an USA national project that extends the concepts of Emulab. The key concept is to build a federation of geographically distributed testbeds to provide users with a strongly heterogeneous infrastructure that will be suitable to a larger variety of networked experiments on dedicated hardware. PlanetLab [9] is a global research network that supports the development of new network services (overlay networks) using virtualization. The topology of PlanetLab is based on a large number ($\tilde{5}00$) sites with 2 or 3 nodes on each site. While it provides a very interesting testbed from the point of view of the distribution of the resources at a global scale for network-based experiments, experiments running at the same time compete for machine-time and network links. Therefore, experiences' reproducibility is not guaranteed, and experiments involving clusters or data centers are not possible. OneLab [6] provides an open federated laboratory, built on PlanetLab Europe, which supports network research for the Future Internet. Finally, FIT [5] from the 2010 French EQUIPEX call targets the Future Internet of Things. It gathers three infrastructures, a cognitive radio testbed, a set of embedded communicating object (ECO) testbeds, and a set of wireless OneLab testbeds mostly designed for various network experiments.

Several Grid targeted platforms also exist along with Grid'5000. DAS-4 [3] is an experimental grid built in the Netherlands. It allows reproducible results but the software stack cannot be configured. FutureGrid [17], which is part of the NSFs TeraGrid high-performance cyber infrastructure in the USA, provides an architecture taking its inspiration from to the one developed in Grid'5000. It targets researches on Grids and Clouds. It increases the capability of the XSEDE to support innovative computer science research requiring access to lower levels of the grid software stack, the networking software stack, and to virtualization and workflow orchestration tools. There is also a large number of production platforms (such as the GENCI supercomputers in France) that are used for different areas of research. They are not mentioned here because the software stack of their clusters cannot be adapted for low- level research experiments or experiments using specific software stacks.

Finally, some platforms allow experiments on Clouds. Amazon EC2/S3 [1] is a commercial Cloud (IaaS platform). While this platform is mainly made for commercial and production applications, several computer science experiments have recently performed on this platform. Google/IBM provided until October 2011 a Cloud running the Hadoop implementation of the MapReduce programming interface. It could be used to test large-scale data application under this protocol. BonFIRE [2] is a FP7 European project supported by the FIRE unit (Future Internet Research and Experimentation) to build a testbed for Internet of Services Experimentation. INRIA is a member of the BonFIRE consortium and one of its 5 testbed providers, thus taking part in the

construction of a European-wide facility for experiment-driven research in Future Internet technologies. Finally, Open Cirrus [7,11] targets experiments around Clouds on bare hardware using distributed clusters available over the world. Led by private companies, it allows multiple experiments using different services (physical resource allocation service, virtual machine resource allocation service, distributed storage service, distributed computing frameworks). VLANs are used to isolate experiments between each others.

6 Conclusions and Future Work

The ability to design and support experiments of large scale distributed algorithms and software is now a mandatory aspect of computer science. When it was started in 2003, the objective of the Grid'5000 project was to ensure the availability of a scientific instrument for experiment-driven research in the fields of large-scale parallel and distributed systems. It has since demonstrated that its fundamental concepts and tools to support experiment-driven research in parallel and distributed systems are solid enough attract a large number of users and to stay pertinent even though the focus of research in these areas has evolved in the past nine years. In the last years, Grid'5000 has had a structuring effect on research in parallel and distributed computing in France. Many French ANR projects have been submitted by Grid'5000 users targeting this platform as their validation instrument. Bridges have been set with production grids. Several collaborations will also be set up with scientists of other disciplines to help them port their applications at a higher scale, exploring new algorithms and parallelization approaches, before using production grids or HPC platforms. Moreover, this platform has been internationally recognized and it serves as a foundation for new scale platforms such as FutureGrid in the US. Hence, Grid'5000 has contributed to solve many challenges in the parallel and distributed computing.

Through our experience in building a large scale and reconfigurable platform and the evolution of researches towards virtualized infrastructures and Clouds, we worked on new features and tools that allow such experiments to be deployed over multiple sites. In this paper, we gave an overview of these tools and the way they can be used for different use cases. However the story is not over and some work remains to be done around new functionnalities.

Whereas abstraction in programming languages enables to design and implement complex IT systems through distributed infrastructures, system virtualization has been mainly limited to one physical machine. With respect to the current utilization of IT through networks in general and Internet in particular, as well as the large amount of available data, the next steps consist in extending virtualization concepts to network and storage facilities. The OpenFlow [8] standard that allows researchers to deploy routing and switching protocols over networks will certainly ease the deployment of large scale network-based experiments. Big Data is also a major research issues for several sciences as well as business applications. Allowing the design of new middleware frameworks for such applications will also require at least new hardware for our experimental platforms (including large number of SSD drives). Finally, we learned that the tools used for the deployment of large scale experiments involving several different software stacks need to be as simple as possible. Simplifying the use of our platform for users is thus also one of our major tasks in the near future.

References

1. Amazon ec2, http://aws.amazon.com/fr/ec2/
2. Bonfire, http://www.bonfire-project.eu/
3. Das-4, http://www.cs.vu.nl/das4/
4. Emulab, http://www.emulab.net/
5. Fit, http://fit-equipex.fr/
6. Onelab, http://www.onelab.eu/
7. Open cirrus, https://opencirrus.org/
8. Openflow, http://www.openflow.org
9. Planetlab, http://www.planet-lab.org/
10. protogeni, http://www.protogeni.net/
11. Avetisyan, A., Campbell, R., Gupta, I., Heath, M., Ko, S., Ganger, G., Kozuch, M., O'Hallaron, D., Kunze, M., Kwan, T., Lai, K., Lyons, M., Milojicic, D., Lee, H.Y., Soh, Y.C., Ming, N.K., Luke, J.Y., Namgoong, H.: Open Cirrus: A Global Cloud Computing Testbed. IEEE Computer 43(4), 42–50 (2010)
12. Booting and using virtual machines on Grid'5000, https://www.grid5000.fr/mediawiki/index.php/Booting_and_Using_Virtual_Machines_on_Grid'5000/
13. Capit, N., Da Costa, G., Georgiou, Y., Huard, G., Martin, C., Mounié, G., Neyron, P., Richard, O.: A batch scheduler with high level components. In: Cluster Computing and Grid 2005 (CCGrid 2005), Cardiff. Royaume-Uni. (2005), http://hal.archives-ouvertes.fr/hal-00005106
14. Cappello, F., Caron, E., Dayde, M., Desprez, F., Jegou, Y., Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Quetier, B., Richard, O.: Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, GRID 2005, pp. 99–106. IEEE Computer Society, Washington, DC (2005), http://dx.doi.org/10.1109/GRID.2005.1542730
15. Claudel, B., Huard, G., Richard, O.: Taktuk, adaptive deployment of remote executions. In: Proceedings of the International Symposium on High Performance Distributed Computing, HPDC (May 2009)
16. Desprez, F., Fox, G., Jeannot, E., Keahey, K., Kozuch, M., Margery, D., Neyron, P., Nussbaum, L., Perez, C., Richard, O., Smith, W., von Laszewski, G., Voeckler, J.: Supporting Experimental Computer Science. Report, Argonne National Laboratory, Argonne (March 2012), http://www.nimbusproject.org/downloads/Supporting_Experimental_Computer_Science_final_draft.pdf
17. FutureGrid, https://portal.futuregrid.org/
18. Jeanvoine, E., Sarzyniec, L., Nussbaum, L.: Kadeploy3: Efficient and Scalable Operating System Provisioning for HPC Clusters. Rapport de recherche RR-8002, INRIA (June 2012), http://hal.inria.fr/hal-00710638
19. Keahey, K., Freeman, T.: Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In: Proceedings of the 2008 Conference on Cloud Computing and Its Applications (CCA), Chicago, IL, USA (2008)
20. Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Elastic management of cluster-based services in the cloud. In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds (ACDC), pp. 19–24. ACM, New York (2009)
21. One-click Cloud deployment tools, https://www.grid5000.fr/mediawiki/index.php/Deployment_Scripts_for_IaaS_Clouds_on_Grid%275000
22. OpenNebula, http://opennebula.org/

23. Opscode. Chef, http://www.opscode.com/chef/
24. Quesnel, F., Lèbre, A., Südholt, M.: Cooperative and Reactive Scheduling in Large-Scale Virtualized Platforms with DVMS. Concurrency and Computation: Practice and Experience, p. XX (December 2012), http://hal.archives-ouvertes.fr/hal-00675315
25. Riteau, P., Tsugawa, M., Matsunaga, A., Fortes, J., Keahey, K.: Large-Scale Cloud Computing Research: Sky Computing on FutureGrid and Grid'5000. ERCIM News (83), 41–42 (2010)
26. Sarzyniec, L., Badia, S., Jeanvoine, E., Nussbaum, L.: Scalability Testing of the Kadeploy Cluster Deployment System using Virtual Machines on Grid'5000. In: SCALE Challenge 2012, Held in Conjunction with CCGrid 2012, Ottawa, Canada (May 2012), http://hal.inria.fr/hal-00700962
27. SC11 Support for Experimental Computer Science Worskhop, http://graal.ens-lyon.fr/~desprez/SC11workshop.htm
28. Shvachko, K., Huang, H., Radia, S., Chansler, R.: The Hadoop distributed file system. In: MSST 2010: Proceedings of the 26th IEEE Symposium on Massive Storage Systems and Technologies, Incline Village, NV, USA, pp. 1–10 (May 2010)
29. The Nimbus Project, http://www.nimbusproject.org/
30. The OAR Project, http://oar.imag.fr/