Enabling Large-Scale Testing of IaaS Cloud Platforms on the Grid'5000 Testbed

Sébastien Badia, Alexandra Carpen-Amarie, Adrien Lèbre, Lucas Nussbaum

Enabling Large-Scale Testing of IaaS Cloud Platforms on the Grid'5000 Testbed

Sébastien Badia
INRIA Nancy - Grand Est
Nancy, France
sebastien.badia@inria.fr

Alexandra Carpen-Amarie
INRIA Rennes - Bretagne Atlantique
Rennes, France
alexandra.carpen-amarie@inria.fr

Adrien Lèbre
Ecole des Mines
Nantes, France
adrien.lebre@inria.fr

Lucas Nussbaum
LORIA / Université de Lorraine
Nancy, France
lucas.nussbaum@loria.fr

ABSTRACT

Almost ten years after its premises, the Grid'5000 platform has become one of the most complete testbeds 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. In this paper, we present the latest mechanisms we designed to enable the automated deployment of the major open-source IaaS cloudkits (i.e., Nimbus, OpenNebula, CloudStack, and OpenStack) on Grid'5000. Providing automatic, isolated and reproducible deployments of cloud environments lets end-users study and compare each solution or simply leverage one of them to perform higher-level cloud experiments (such as investigating Map/Reduce frameworks or applications).

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems, Cloud Computing; B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; K.6.2 [**Management of Computing and Information Systems**]: Installation Management

General Terms

Experimentation, Design, Management, Measurement, Performance

Keywords

Cloud computing, IaaS, virtualization, Grid'5000, reproducibility, large-scale experiments, isolation, OpenNebula, OpenStack, Nimbus, Cloudstack

TTC '13, July 15, 2013, Lugano, Switzerland

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 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 an 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, and second, because of the inability to create controlled experimental conditions, and to deal with the wide 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 [5], Grid'5000 was solid enough to attract more than 600 users and led to a large number of research results and publications.

Considering the phenomenal impact of Cloud Computing solutions over the last five years, it became essential for the Grid'5000 consortium to extend the testbed in order to be able to study and address Cloud Computing concerns. Among them, Quality of Service (QoS), fault-tolerance, energy management, and scalability are a few major ones. Extending the Grid'5000 software and services stack to investigate such issues is a critical aspect for the community, as experimenting with cloud platforms in a rigorous and scientific manner is rather difficult. Lacking access to open and configurable cloud platforms, a large number of scientific experiments have been and are performed on the Amazon EC2/S3 [9] commercial Infrastructure-as-a-Service Cloud, where the isolation and reproducibility criteria are simply impossible to achieve.

Several Grid-targeted platforms were also developed along with Grid'5000. DAS-4 [8] is an experimental grid built in the

Netherlands. It allows reproducible results but the software stack cannot be configured. FutureGrid [10], which is part of the NSF's TeraGrid high-performance cyber infrastructure in the USA, provides an architecture taking its inspiration from to the one developed in Grid'5000. It targets research on Grids and Clouds, providing access to lower levels of the grid software stack, the networking software stack, and to virtualization and workflow orchestration tools. Additionally, a large number of production platforms (such as the GENCI supercomputers in France) are used for different areas of research. Typically, their software stacks cannot be adapted for low-level experiments and the end users are not allowed to customize the environment installed on the machines.

In this paper, we describe the latest contributions of the Grid'5000 software and services stack to enable large-scale experiments involving new IaaS (Infrastructure-as-a Service) cloud technologies. These contributions make Grid'5000 one the very few platforms that allow the orchestration of such experiments between multiple sites and in an isolated and reproducible manner. The Open Cirrus [6, 2] platform targets a similar objective around clouds on bare hardware using distributed clusters available over the world. Led by private institutions, it allows multiple experiments using different services (physical resource allocation service, virtual machine resource allocation service, distributed storage service, distributed computing frameworks). However, as far as we know, the software stack is not as advanced as the one we propose.

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 enabling the use of virtualization technologies at the level of the Grid'5000 software and service stack. Section 4 focuses on deployment tools for open-source IaaS environments, designed to install fully-functional clouds on the Grid'5000 testbed and to tune them according to user requirements. Finally, Section 5 concludes the article.

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 [5] 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, con-

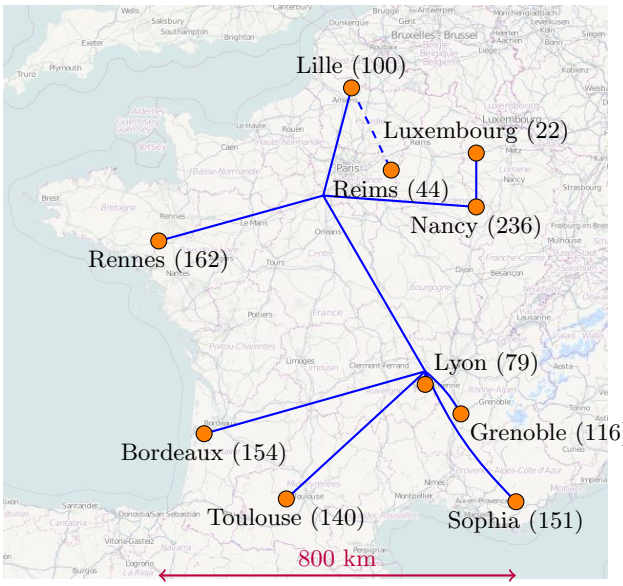

Figure 1: Grid'5000 sites and their number of nodes.

nections 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 processes 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 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 [4, 22]. Users can install their own system image on the nodes (without any virtualization layer) using Kadeploy [11]. Experiments requiring network isolation can

use KaVLAN to reconfigure switches and isolate nodes from the rest 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 utilize the preconfigured environment on the 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 employed 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 as maintenance events become frequent when 10 sites are concurrently operated.

3. A SOFTWARE STACK FOR CLOUD EXPERIMENTS

This section describes three key Grid'5000 services that contribute to support 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 switch reconfiguration).

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 limitation for experimentation, since users often need to assess the scope of the obtained results, verifying that 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 feature allows users to perform experiments without being bound to one particular Linux distribution or version, or even operating system.

While it is common for cloud infrastructures to provide the ability to deploy custom OS images in virtual machines, Grid'5000 implements 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 the *Kadeploy* [11] 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 [7]; chain-based image broadcast [11]). The deployment process is controlled by an automaton that handles the unavoidable errors (caused by unreliable protocols and hardware), and the corresponding retry policies. Due to these 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 customized environments.

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 [22]. Users can reserve a set of network IP addresses (from /22 to a /16) at the same time as nodes. As an example,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

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

VLAN type	Ethernet isolation	IP isolation	Multi-site
local	yes	no	no
routed	yes	no	no
global	yes	no	yes

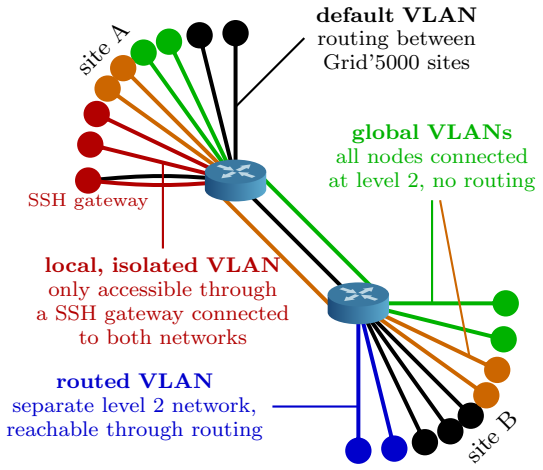

Figure 2: Types of VLAN provided by *KaVLAN*.

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 of 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 through a machine connected to both the VLAN and the testbed's network (generally by using SSH).
- **Routed VLAN** also provides users with a separate L2 network. The network is however not isolated at the L3 level, thus being reachable from any node of the testbed. 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 through routing.

4. DELIVERING READY-TO-USE CLOUD PLATFORMS ON GRID'5000

The first step towards investigating Infrastructure-as-a-Service environments within Grid'5000 was achieved through a set of "sky computing" experiments [20]. They involved a federation of several Nimbus Clouds [12] spanning across Grid'5000 and FutureGrid [10] and harnessing over 1500 cores for MapReduce applications. These experiments showed that testbeds such as Grid'5000 may play an essential role in enabling experimental research at all levels of the Cloud Computing stack and providing 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, we developed a set of deployment tools for open-source IaaS environments, capable of installing and tuning fully-functional clouds on the Grid'5000 testbed [15]. The deployment tools support four widely-used IaaS clouds, namely OpenNebula [13, 17], CloudStack [1], Nimbus [12, 21] and OpenStack [18]. This section provides insights on the design of these tools, detailing the cloud-specific issues that drove their development. We introduce two approaches for providing cloud platforms to Grid'5000 users, based on the networking requirements and limitations of each cloud.

4.1 One-Click IaaS Clouds with g5k-campaign

This section describes a cloud deployment utility that makes use of *g5k-subnets* to provision the virtual IP addresses used within the cloud. While simple and flexible, this approach poses some limitations on the networking configuration of the cloud environment, being available only for platforms that can accommodate IP ranges served by external DHCP servers. As OpenNebula, CloudStack and Nimbus support such configurations, we designed a generic deployment utility to install and configure any one of these cloud platforms over Grid'5000.

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 experiment description files called *engines*. An engine represents the implementation of a Grid'5000 experiment and defines all of its stages: physical node reservations in Grid'5000, environment deployment, configuration, and experiment execution. We developed specific engines for each cloud framework, which can be activated through easy-to-use configuration files passed to the *g5k-campaign* framework. Thus, users can supply customized requirements for each stage of the Grid'5000 experiment execution, ranging from Grid'5000 node reservation constraints to cloud deployment settings.

Implementation.

The deployment tools are designed to support a wide range of cloud-specific parameters, which are either automatically configured with default values or customized by the user. Such parameters include hypervisor and virtualization settings, host nodes configuration, installation of external packages, authentication settings, virtual networks creation, configuration of various storage mechanisms for VM images and of cloud user interfaces, cloud services initialization.

The implementation of the 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. The deployment tools are written in Ruby and the installation and configuration are done on each physical node by using the Chef [19] configuration management framework. The Chef cookbooks are designed in a modular manner, to allow users to add or extend the current configuration options.

We validated these tools by installing each cloud platform on tens of physical nodes capable of executing hundreds of virtual machines. As an example, the OpenNebula environment was tested on 80 physical nodes belonging to 3 Grid'5000 sites. On this platform we deployed a virtual cluster comprising 350 VMs that was used to run Hadoop applications. 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.

Zoom on the cloud deployment engine.

The deployment tools rely on cloud deployment engines for each specific cloud platform. Each such engine is responsible for handling the installation process of the cloud environment, either from Linux packages or from specific source code archives. It automatically carries out the deployment and configuration, featuring two types of installation modes. The first one is fully automatic and does not require any specific configuration from the user. It thus enables users without any knowledge on the internal design of the cloud platform to use it through standard interfaces and to run cloud applications in a similar fashion to public clouds. The second one allows users to customize the various cloud services to meet their needs.

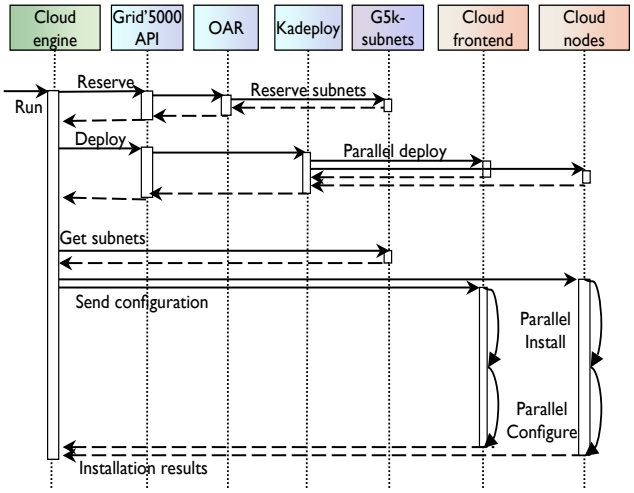

Figure 3: Sequence diagram of a cloud deployment engine execution.

A cloud engine can be executed by passing a configuration file to the *g5k-campaign* tool, which is in charge of interpreting it and delivering the ready-to-use cloud platform. The sequence diagram in Figure 3 describes the execution workflow of each such engine. First, a node reservation is made for each site specified in the configuration file through the *Grid'5000 API*, along with *g5k-subnets*-based ranges 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*. In each cloud platform, a node can play one of the following roles: cloud frontend, compute node or client node. Once the nodes are operational, the cloud engine retrieves the reserved IP ranges from each site and then creates specific configuration settings for each node, according to their role. Finally, the selected cloud is installed and configured on each node in parallel. 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 is configured with several virtual networks, each of them corresponding to a group of host nodes belonging to the same site.

4.2 OpenStack in FlatDHCP mode thanks to KaVLAN

While it is useful and proved to work very well, the approach described in the previous section is limited when applied to more complex cloud stacks or configurations.

Specifically, the recommended way to use OpenStack (until its *folsom* release) is `nova-network`'s *FlatDHCP* mode, where a DHCP server is started and managed by OpenStack. This

mode is not usable with *g5k-subnets* reservations, as the OpenStack DHCP server would conflict with the testbed's DHCP server.

To alleviate this problem, we designed another approach, using a KaVLAN routed VLAN to host the OpenStack cloud and its own specific network environment. Connections to the outside world are made either through the VLAN gateway, or via special nodes that have two network interfaces (one on the KaVLAN network, the other one on the standard network). This approach avoids all possible perturbations between OpenStack and the rest of the testbed.

This approach also differs in terms of software used to automate the setup of the cloud infrastructure. The previous approach used custom-made Chef cookbooks. In the case of OpenStack, we used the official Puppet recipes, as provided by PuppetLabs (and more recently, StackForge). This facilitates the maintenance, as the maintenance effort is shared with other OpenStack users, especially when new OpenStack releases are made available. It also eases the sharing with the community, since the resulting OpenStack deployment has less specificities.

We validated our tools by deploying an OpenStack cloud on about 30 physical machines in less than 20 minutes. Our cloud included one Cloud Controller (API, network, scheduler, dashboard, object store, database, authentication, etc.), and all the other nodes hosting virtual machines (`nova-compute`) and providing block storage with Cinder. The deployment of OpenStack on Grid'5000 was used as a staging area to port a bioinformatics data-oriented workflow to AWS.

5. CONCLUSION

The ability to design and support evaluations 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 are solid enough to attract a large number of users and to be internationally recognized. As large-scale platforms evolved towards virtualized infrastructures and clouds, we enhanced the Grid'5000 reconfigurable testbed with new features and tools that allow such experiments to be deployed over multiple sites. In this paper, we gave an overview of these tools. However the story is not over and some work remains to be done around new functionnalities. A first challenge is to extend the Grid'5000 software stack to completely virtualize the network abstractions. By leveraging technologies implementing the OpenFlow [16] standard, end-users will be able to study new concerns around the *Network-as-a-Service* paradigm and propose new routing and switching protocols to ensure the QoS of network communications. Big Data is also a major research issue where end-users expect a lot from cloud computing platforms. Allowing the design of new middleware frameworks for such applications require at least new hardware for our experimental platforms (including a large number of SSD drives, currently not available on the Grid'5000 testbed). 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 one of our major tasks in the near future as well. Furthermore, we plan to enhance the cloud deployment tools with support for other IaaS frameworks, such as Eucalyptus [14], and to investigate their capabilities for hosting Platform-as-a-Service offerings. The background of the scientific and the technical boards of Grid'5000 is an important advantage in the design of new facilities dedicated to distributed system experimentations. Supported by the FIRE unit (Future Internet Research and Experimentation), several Grid'5000 members take part to the FP7 European BonFIRE project [3] that aims at the construction of a European-wide facility for experiment-driven research in Future Internet technologies.

6. ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several universities as well as other funding bodies (see https://www.grid5000.fr).

7. REFERENCES

[1] Apache CloudStack. http://cloudstack.apache.org/.
[2] A. Avetisyan, R. Campbell, I. Gupta, et al. Open Cirrus: A Global Cloud Computing Testbed. *IEEE Computer*, 43(4):42–50, Apr. 2010.
[3] BonFire. http://www.bonfire-project.eu/.
[4] N. Capit, G. Da Costa, Y. Georgiou, et al. A batch scheduler with high level components. In *Cluster computing and Grid 2005 (CCGrid'05)*, 2005.
[5] F. Cappello, E. Caron, M. Dayde, et al. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In *Grid'05*, 2005.
[6] O. Cirrus. https://opencirrus.org/.
[7] B. Claudel, G. Huard, and O. Richard. Taktuk, adaptive deployment of remote executions. In *In Proc. of the Int. Symposium on High Performance Distributed Computing (HPDC)*, May 2009.
[8] DAS-4. http://www.cs.vu.nl/das4/.
[9] A. EC2. http://aws.amazon.com/ec2/.
[10] FutureGrid. https://portal.futuregrid.org/.
[11] E. Jeanvoine, L. Sarzyniec, and L. Nussbaum. Kadeploy3: Efficient and Scalable Operating System Provisioning for HPC Clusters. Rapport de recherche RR-8002, INRIA, June 2012.
[12] K. Keahey and T. Freeman. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In *Proc. of the 2008 Conf. on Cloud Computing and Its Applications (CCA)*, USA, 2008.
[13] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Elastic management of cluster-based services in the cloud. In *Proc. of the 1st Workshop on Automated control for datacenters and clouds (ACDC)*, pages 19–24, 2009.
[14] D. Nurmi, R. Wolski, C. Grzegorczyk, et al. The Eucalyptus Open-Source Cloud-Computing System. In *Proc. of the 9th Int. Symposium on Cluster Computing and the Grid (CCGRID)*, pages 124–131, USA, 2009.
[15] One-click Cloud deployment tools. https://www.grid5000.fr/mediawiki/index.php/Deployment_Scripts_for_IaaS_Clouds_on_Grid%275000.
[16] OpenFlow. http://www.openflow.org.
[17] OpenNebula. http://opennebula.org/.
[18] OpenStack. http://www.openstack.org/.
[19] Opscode. Chef. http://www.opscode.com/chef/.
[20] P. Riteau, M. Tsugawa, A. Matsunaga, et al. Large-Scale Cloud Computing Research: Sky Computing on FutureGrid and Grid'5000. *ERCIM News*, (83):41–42, Oct. 2010.
[21] The Nimbus Project. http://www.nimbusproject.org/.
[22] The OAR Project. http://oar.imag.fr/.