Resources Description, Selection, Reservation and Verification on a Large-scale Testbed

David Margery, Emile Morel, Lucas Nussbaum, Olivier Richard, Cyril Rohr

Resources Description, Selection, Reservation and Verification on a Large-scale Testbed

David Margery[1], Emile Morel[1], Lucas Nussbaum[2], Olivier Richard[3], and Cyril Rohr[1]

[1] Inria
[2] LORIA - Université de Lorraine, Inria, CNRS
[3] LIG - Université de Grenoble, Inria, CNRS

firstname.lastname@inria.fr

Abstract. The management of resources on testbeds, including their description, reservation and verification, is a challenging issue, especially on of large scale testbeds such as those used for research on High Performance Computing or Clouds. In this paper, we present the solution designed for the Grid'5000 testbed in order to: (1) provide users with an in-depth and machine-parsable description of the testbed's resources; (2) enable multi-criteria selection and reservation of resources using a HPC resource manager; (3) ensure that the description of the resources remains accurate.

Key words: testbed; resources; discovery; description; reservation; verification

1 Introduction

All scientific domains relying on experimental methodology require testbeds: particle colliders, synchrotrons, telescopes, greenhouses and experimental farms, etc. Computer science is not different, and the availability of testbeds is a requirement for solid science in fields such as distributed systems and networking.

When working on distributed systems such as Cloud computing infrastructures, high performance computing (HPC), peer-to-peer systems, or grid infrastructures, the scalability of solutions is often of central importance. To evaluate it, researchers rely on large-scale testbeds, much larger than what one laboratory or organization can usually fund and host, and often composed of a large number of resources scattered over various geographical locations.

Such testbeds and their resources raise a number of challenges, from both a testbed operator's and a user's point of view. Specifically, testbed operators need to keep track of all available resources, expose information about them to users, and make sure that they are still functioning adequately. Conversely, users need to be able to explore the resources and reserve them according to their experimental needs. Those operations have a critical role in the quality of

research performed on such testbeds, both by ensuring that the initial experimental results are not tainted by malfunctioning or misdocumented resources, and by enabling the experimenter to describe the experimental environment in a way suitable for reproducible research.

This paper describes the framework designed in the context of the Grid'5000 testbed [7, 5], a large scale testbed focusing on Cloud, HPC, P2P and Grid computing, to address the challenges of resources description, selection, reservation and verification. The remainder of this paper is organized as follows. Section 2 describes the Grid'5000 testbed in order to provide the context of this work. Section 3 describes our multi-tier framework for resources management. Related work is discussed in Section 4. Finally, we conclude that paper in Section 5.

2 Context: the Grid'5000 testbed

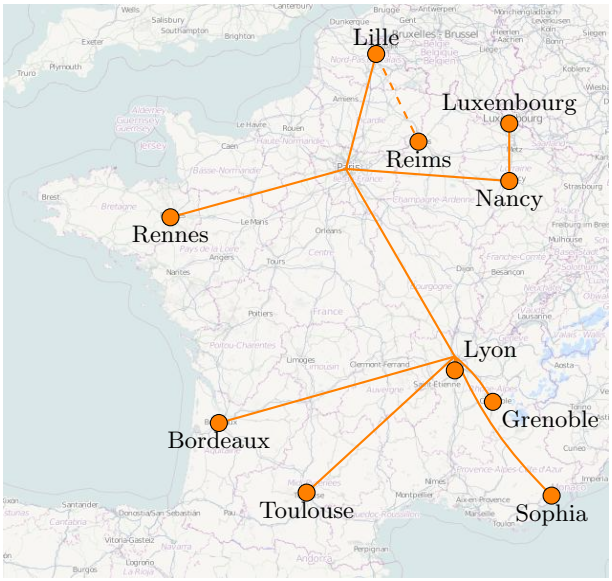

Fig. 1. Map of the Grid'5000 testbed.

Started in 2003 and opened to the general public in 2004, the Grid'5000 project aims at providing a testbed for research on parallel and distributed systems, enabling researchers to validate their theoretical results and their software developments. It has been used for a large number of experiments, ranging from small scale to very large scale, and on a variety of topics: low level network protocols, virtualization environments, applications, etc. Grid'5000 has had between 500 and 600 active users per year since 2009, and at least 328 publications have been using Grid'5000.

As of 2014, Grid'5000 is composed of 11 sites (Figure 1), 26 clusters, 1260 nodes, and 8000 CPU cores. One particular focus of the testbed is the level of reconfiguration available to users: they can deploy their own software stacks (operating system, applications) on the bare metal using Kadeploy [10], with no particular constraint. They can also reconfigure the network to isolate their experiments on the ethernet level (VLAN), in order to protect it from external perturbations, or to protect the testbed from intrusive protocols involved in the experiment. Thanks to those reconfiguration capabilities, Grid'5000 enables users to work on Cloud or Grid middlewares (OpenStack, OpenNebula, Nimbus, gLite) for the duration of an experiment.

3 Resources management in Grid'5000

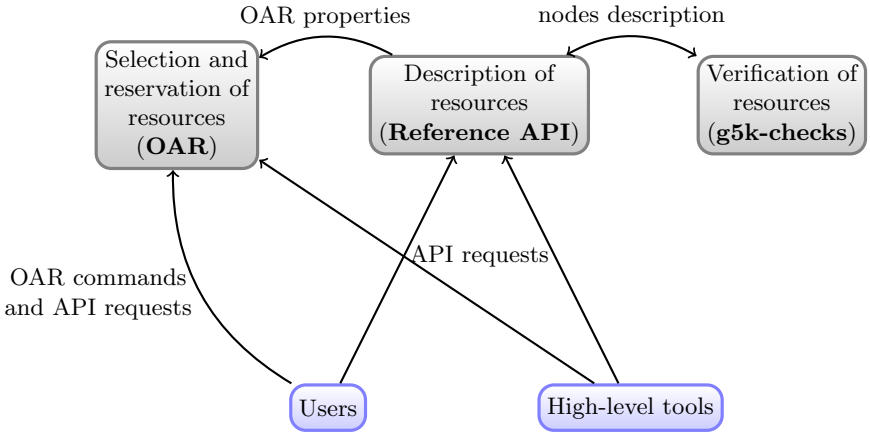

Fig. 2. General structure of resource management framework.

Figure 2 provides a general overview of our framework designed in the context of the Grid'5000 project to manage resources. This framework is composed of three components:

- The *Reference API* (Section 3.1) contains the description of all resources, as a set of JSON documents.
- The *OAR resource manager* (Section 3.2) enables users to select and reserve resources.
- The *g5k-checks* tool (Section 3.3) is in charge of the verification of resources.

3.1 Resources description with the Reference API

All resources descriptions are centralized in a NoSQL database, as a set of JSON documents that can be retrieved through a RESTful API. Those documents

```
"supported_job_types" : {
   "deploy" : true,
   "besteffort" : true,
   "virtual" : "ivt"
},
"chassis" : {
   "serial" : "27Q7NZ1",
   "manufacturer" : "Dell Inc.",
   "name" : "PowerEdge R720"
},
"bios" : {
   "version" : 2,
   "release_date" : "08/29/2013",
   "vendor" : "Dell Inc."
},
"architecture" : {
   "platform_type" : "x86_64",
   "smp_size" : 2,
   "smt_size" : 16
},
"processor" : {
   "instruction_set" : "x86-64",
   "cache_l1i" : 32768,
   "version" : "E5-2650",
   "cache_l2" : 262144,
   "model" : "Intel Xeon",
   "cache_l1d" : 32768,
   "cache_l3" : 20971520,
   "vendor" : "Intel",
   "clock_speed" : 2000000000
},
"main_memory" : {
   "ram_size" : 270991937536,
},
"storage_devices" : [
   {
      "rev" : "DL10",
      "model" : "INTEL SSDSC2BB30",
      "interface" : "SATA II",
      "device" : "sda",
      "size" : 300069052416,
      "driver" : "megaraid_sas"
   },
   {
      "rev" : "DL10",
      "model" : "INTEL SSDSC2BB30",
      "interface" : "SATA II",
      "device" : "sdb",
      "size" : 300069052416,
      "driver" : "megaraid_sas"
   }
],
```

```
"network_adapters" : [
   {
      "ip" : "172.16.68.1",
      "rate" : 10000000000,
      "mountable" : true,
      "interface" : "Ethernet",
      "mounted" : true,
      "mac" : "b8:ca:3a:69:12:68",
      "enabled" : true,
      "version" : "82599EB",
      "device" : "eth0",
      "ip6" : "fe80::baca:3aff:fe69:1268",
      "network_address" : "graphite-1",
      "switch_port" : "F1",
      "switch" : "gw-nancy",
      "management" : false,
      "driver" : "ixgbe",
      "vendor" : "intel"
   },
   {
      "version" : "IDRAC7",
      "ip" : "172.17.68.1",
      "device" : "bmc",
      "network_address" : "graphite-1-bmc",
      "switch_port" : "1/0/41",
      "rate" : 100000000,
      "switch" : "sgraphene3-ipmi",
      "mountable" : false,
      "interface" : "Ethernet",
      "mounted" : false,
      "mac" : "f0:1f:af:e1:9a:0c",
      "management" : true,
      "vendor" : "DELL",
      "enabled" : true
   }
],
"sensors" : {
   "power" : { "via" : { "pdu" : {
            "uid" : "graphene-pdu9",
            "port" : 24
         },
         "api" : { "metric" : "pdu" }
      },
   }
},
"mic" : {
   "mic_model" : "7120P",
   "mic" : true,
   "mic_count" : 1
},
"performance" : {
   "core_flops" : 13170000000,
   "node_flops" : 187900000000
}
```


Fig. 3. Description of one node in the Reference API.

provide an in-depth description of most of the testbed's resources: nodes of course (Figure 3), but also network equipment (switches, routers) and other equipment such as power distribution units (PDU), with information such as the vendor, product name and reference, how the equipment is connected, and instructions on how to access remote control and measurement capabilities through SNMP.

This API is used by a number of tools, to generate documentation (list of all available resources), maps (e.g. network topology), and to enable remote control of resources. For example, the KaVLAN tool in charge of network isolation using VLAN reconfiguration on switches, relies on the mapping between nodes and network switches provided by the Reference API.

Most of the data in the Reference API is collected automatically by *g5k-checks*, which will be described in Section 3.3. A few data fields, whose value cannot be automatically discovered, require manual intervention from the testbed operator.

The data stored in the reference API is archived in a Git repository, which makes it possible to follow the testbed's evolutions over time, or retrieve the state of the testbed at a given date, which is useful in order to explain specific experimental results.

3.2 Resources selection and reservation with OAR

Reserving two nodes for two hours. Nodes must have a GPU and power monitoring:

```
oarsub -p "wattmeter='YES' and gpu='YES'" -l nodes=2,walltime=2 -I
```

Reserving one node on cluster a, and two nodes with a 10 Gbps network adapter on cluster b:

```
oarsub -I -l "{cluster='a'}/nodes=1+{cluster='b' and eth10g='YES'}/nodes=2,walltime=2"
```

Advance reservation of 10 nodes on the same switch with support for Intel VT (virtualization):

```
oarsub -l "{virtual='ivt'}/switch=1/nodes=10,walltime=2" -r '2014-11-08 09:00:00'
```

Fig. 4. Example uses of OAR properties to request resources matching requirements.

As the Grid'5000 infrastructure was originally designed by the HPC community, it felt natural at the time to use a HPC batch scheduler as the building block for resource reservation. The OAR resource manager [6, 1] provides two main useful features in the context of a testbed.

Resources properties Most HPC resource managers such as Slurm [13] limit the structuring of resources in *partitions* of nodes considered identical. On the

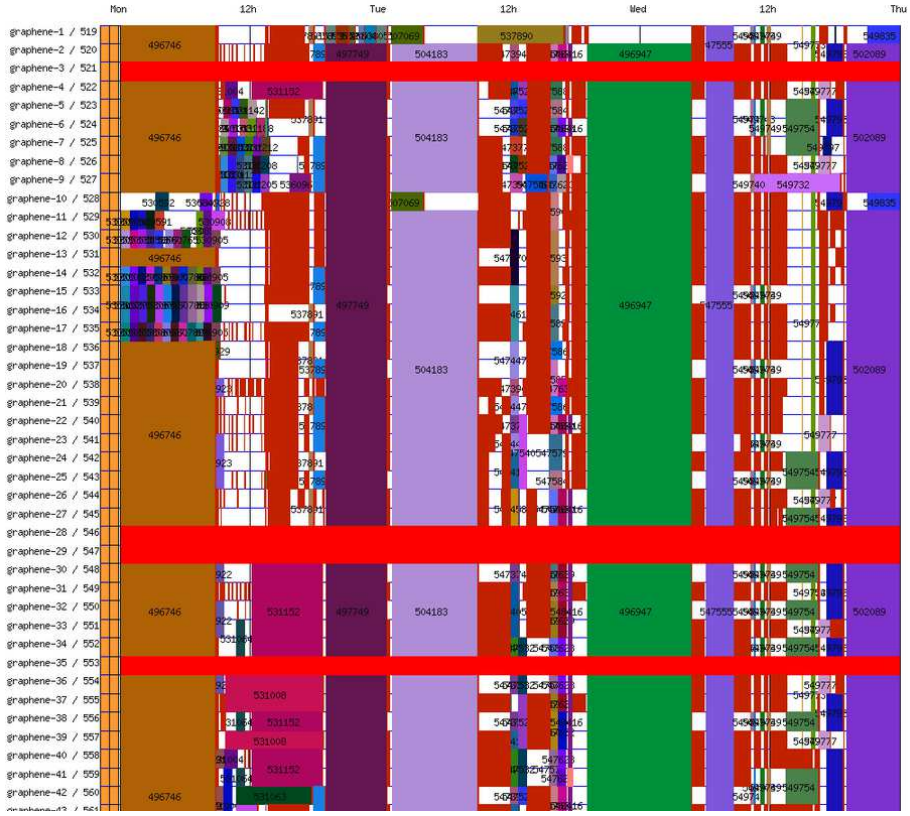

Fig. 5. Visualization of the usage of a Grid'5000 site using a Gantt diagram.

other hand, with OAR, each node can be associated with a number of properties, that can later be used to request resources fulfilling specific needs, as shown in Figure 4. This enables users to request resources matching specific properties, rather than rely on explicit naming of requested resources: the burden of filtering resources for specific user requirements is transferred from the user to the resource manager.

On Grid'5000, OAR properties are generated automatically using data from the *Reference API*, and cover most of the data available in it. Also, resources are not limited to nodes. On Grid'5000, we also use OAR to reserve storage space and network subnets (for experiments requiring additional network addresses for virtual machines).

Advance reservations of resources Most HPC resource managers focus on *batch* tasks, where the resource manager tries to start tasks as early as possible given available resources, but does not provide an opportunity for the user to specify when the resources should be allocated. This is a reasonable strategy for a HPC cluster, as the tasks do not require any user intervention. However,

in the context of a testbed, most experiments require human intervention, as experiments that can run automatically without problems are quite rare.

OAR provides a *batch* scheduling mode, but also provides the possibility to request *advance reservation* of resources: at the time of the resource request, users can specify when they would like the resources to be allocated, and for which duration. On Grid'5000, this is coupled with a policy of only allowing large reservations of resources during nights and week-ends: during week days, this favors shared use of the testbed by many users so that they can prepare their experiments, and then run the actual experiments during nights and week-ends. OAR also provides a visualization of the planned reservations as a Gantt diagram (Figure 5).

3.3 Resources verification with *g5k-checks*

A key challenge when maintaining a description of a testbed is to ensure that it is accurate. Inaccuracies could mislead research into making false assumptions, with dramatic consequences: incorrect results, publications that need to be retracted, etc. Unfortunately, hardware failures that have an impact on experiments are not uncommon: malfunctioning hard disk drives or broken RAM memory banks that result in less memory being available occur on a weekly, if not daily basis on Grid'5000.

On Grid'5000, we designed a tool called *g5k-checks*. *g5k-checks* is run when a node boots, and at regular intervals when no jobs are scheduled on a node. It is in charge of verifying that the node matches its description in the Reference API, by: (1) retrieving the current description of the node; (2) using several tools to acquire information on the node; (3) comparing the acquired information with the one from the Reference API, and marking the node as unavailable if the information does not match.

For most of its work, *g5k-checks* uses Ohai [2], a Ruby library to detect various attributes on a node. Additionally, it also relies on tools such as `ethtool` to gather the configuration of network interfaces.

g5k-checks has two additional modes of operation. First, it can be run directly by users during jobs where they pushed their own software environment to the nodes, which is useful in order to record the execution environment of a job. Second, testbed operators can use it to collect the information needed to fill the Reference API. This procedure is also a good opportunities to detect outliers among nodes, which are not uncommon when one installs a large new cluster.

3.4 Interfaces with external tools

During the design of this framework, a lot of focus has been put in enabling external contributors to develop their own tools. Both the Reference API and OAR provide APIs that can be used to automate complex tasks. For example, one limitation of the Grid'5000 testbed is its distributed nature: each site has its own instance of the resource manager, which can make it hard to reserve a large

number of resources on several sites at once. So one Grid'5000 user developed a tool, called Funk, that uses the Reference and OAR API to find resources matching a specification, and analyze the planning of reservations in order to find a matching time slot.

4 Related work

Despite a large number of testbeds, both for Internet of Things and WSN [9] and for HPC and Cloud [8], our work is the only one presented as an integrated solution for the management of resources – most of the testbeds seem to only address a subset of the facets addressed in that paper.

Resource description and selection has been the focus on the more attention. Several solutions have been designed over the years, such as PlanetLab's SWORD [11], GENI's RSpec [3] or SensorML [4].

Most testbeds, especially in the WSN context, do not provide a way to reserve resources in advance. A notable exception is SensLab [12], which also uses OAR, and was actually inspired by our work on Grid'5000 (as both testbeds are mainly located in France).

Finally, while many tools can be used to explore the content of a resource (`lshw`, `hwinfo`) or to benchmark it, we are not aware of any other general attempt of comparing the results of such tools with a reference in order to detect malfunctioning hardware in the context of a testbed.

5 Conclusions and future work

In this paper, we presented an integrated, and fully-functional solution for the management of resources designed in the context of the Grid'5000 testbed. This solution provides users with an in-depth and machine-parsable (JSON) description of the testbed's resources. It enables multi-criteria selection and reservation of resources by using the OAR HPC job scheduler. Finally, it ensures that the description of resources remains accurate, by comparing it on a regular basis with information retrieving by a tool executed on each resource.

The main area of future improvement of this work is the verification of resources. Most of the tests currently implemented confirm the presence and the vendor & product information of a given piece of hardware, but do not verify its performance. It would be extremely useful to extend *g5k-checks* with some performance testing, especially for pieces of hardware that tend to fail frequently, such as hard disk drives. However, this needs to be balanced with the load imposed on the testbed by such testing – one need to use or design testing tools that are sufficiently efficient to provide an accurate performance measurement after a very short test.

References

1. OAR - a versatile resource and task manager for hpc clusters and other computing infrastructures. http://oar.imag.fr/
2. Ohai. http://docs.opscode.com/ohai.html
3. Rspec. http://www.protogeni.net/wiki/RSpec
4. Aloisio, G., Conte, D., Elefante, C., Epicoco, I., Marra, G.P., Mastrantonio, G., Quarta, G.: Sensorml for grid sensor networks. In: GCA. pp. 147–152 (2006)
5. Balouek, D., Amarie, A., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., Lèbre, A., Margery, D., Niclausse, N., Nussbaum, L., Richard, O., Perez, C., Quesnel, F., Rohr, C., Sarzyniec, L.: Adding virtualization capabilities to the grid'5000 testbed. In: Ivanov, I., Sinderen, M., Leymann, F., Shan, T. (eds.) Cloud Computing and Services Science, Communications in Computer and Information Science, vol. 367, pp. 3–20. Springer International Publishing (2013), http://dx.doi.org/10.1007/978-3-319-04519-1_1
6. Capit, N., Da Costa, G., Georgiou, Y., Huard, G., Martin, C., Mounie, G., Neyron, P., Richard, O.: A batch scheduler with high level components. In: International Symposium on Cluster Computing and the Grid (CCGrid 2005). vol. 2, pp. 776–783 Vol. 2 (May 2005)
7. Cappello, F., Desprez, F., Dayde, M., Jeannot, E., Jégou, Y., Lanteri, S., Melab, N., Namyst, R., Primet, P., Richard, O., Caron, E., Leduc, J., Mornet, G.: Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform. In: 6th IEEE/ACM International Workshop on Grid Computing (Grid). pp. 99–106 (Nov 2005), http://hal.inria.fr/inria-00000284/en/
8. Desprez, F., Fox, G., Jeannot, E., Keahey, K., Kozuch, M., Margery, D., Neyron, P., Nussbaum, L., Pérez, C., Richard, O., Smith, W., Von Laszewski, G., Vöckler, J.: Supporting Experimental Computer Science. Rapport de recherche Argonne National Laboratory Technical Memo 326 (Mar 2012), http://hal.inria.fr/hal-00720815
9. Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., Razafindralambo, T.: A survey on facilities for experimental internet of things research. Communications Magazine, IEEE 49(11), 58–67 (2011)
10. Jeanvoine, E., Sarzyniec, L., Nussbaum, L.: Kadeploy3: Efficient and Scalable Operating System Provisioning. USENIX ;login: 38(1), 38–44 (Feb 2013)
11. Oppenheimer, D., Albrecht, J., Patterson, D., Vahdat, A.: Distributed resource discovery on planetlab with SWORD. In: Proceedings of the ACM/USENIX Workshop on Real, Large Distributed Systems (WORLDS) (2004)
12. des Roziers, C.B., Chelius, G., Ducrocq, T., Fleury, E., Fraboulet, A., Gallais, A., Mitton, N., Noël, T., Vandaele, J.: Using SensLAB as a first class scientific tool for large scale wireless sensor network experiments. In: NETWORKING 2011, pp. 147–159 (2011)
13. Yoo, A.B., Jette, M.A., Grondona, M.: Slurm: Simple linux utility for resource management. In: Job Scheduling Strategies for Parallel Processing. pp. 44–60. Springer (2003)