Grid5000: An Experimental Grid platform for Computer Science

Yiannis Georgiou and Olivier Richard
MESCAL, Laboratoire Informatique et Distribution (ID)-IMAG
ZIRST 51, avenue Jean Kuntzmann 38330 Montbonnot Saint Martin - FRANCE,
{Yiannis.Georgiou|Olivier.Richard}@imag.fr

Abstract

Research upon Computer Science, especially in large scale distributed systems like Grids, P2P systems and High Performance Computing (HPC) areas, has to deal with issues related to increasingly complex systems. Theoretical analysis, simulation and even emulation seem not adequate enough for a complete study of these systems. Hence the need for new generation of scientific instruments capable for the observation of complex distributed systems running at real scale and under reproducible experimental conditions has arisen. Grid5000 has been designed to answer to this need. It provides a real-life experimental research tool for computer scientists. In this paper we discuss the importance of an experimental grid platform dedicated to Computer Science research. We present the design choices of Grid5000 architecture and we analyze the key components of the platform which is OAR Resource and Job Management System, Kadeploy reconfiguration toolkit along with all the monitoring and experiments steering tools.

1 Introduction and Motivations

Research in Grids, P2P systems and High Performance Computing is based on a variety of methodologies and tools. When Grid5000 [1, 2] was designed, most of the research conducted in Grids and P2P systems was performed using simulators, emulators or production platforms. However, all these tools present limitations making the study of new algorithms and optimizations difficult. Simulators focus on a specific behavior or mechanism of the distributed system and abstract the rest of the system. Hence not all factors and conditions that influence the distributed system can be fully experimented. On the other side, emulators provide a tool capable for executing the actual software of the distributed system, in its whole complexity. Nevertheless, they fail to capture all the dynamic, variety and complexity of real life conditions. Production platforms could provide a good solution for real-life experimentation. However, the fact that specific experiments need specialized software which is often hard to install on production platforms along with the big difficulty of experiment reproduction are some important limitations.

Hence, the complexity of Grid and P2P systems raise the need for real-scale experimental platforms where computer scientists can run experiments, observe the distributed systems behavior at large scale under real-life conditions and make precise measurements.

As a matter of fact, the experiments upon complex distributed systems span over all the layers of the software stack between the user and the hardware (figure 1). The applications, programming environment, runtime systems, middleware, operating systems and networking layers are subject to extensive studies seeking to improve their performance, security, fairness, robustness and quality of service. Thus, a mechanism that would facilitate the experimentation upon all different layers of the software stack became indispensable.

Grid5000 was designed to answer to the above needs. It is a large-scale distributed platform that can be easily controlled, reconfigured and monitored. Especially designed for computer science, it provides a real-life experimental tool, ideal for research upon complex distributed systems.

Many institutes and international programs have developed various tools to foster large-scale distributed systems research. Platforms like PlanetLab [3], Emulab [4], GENI [5] and DAS [6] provide some significant examples. The main difference of Grid5000 with all these platforms is the degree of reconfigurability. This functionality, allows researchers to deploy and install the exact software environment they need for their experiments, making the platform an ideal tool for real-

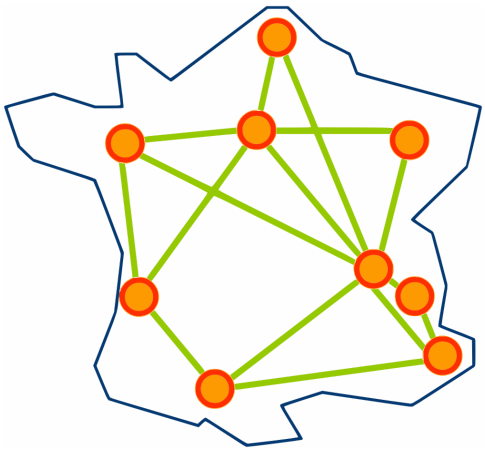

Figure 1. Grid5000 national grid.

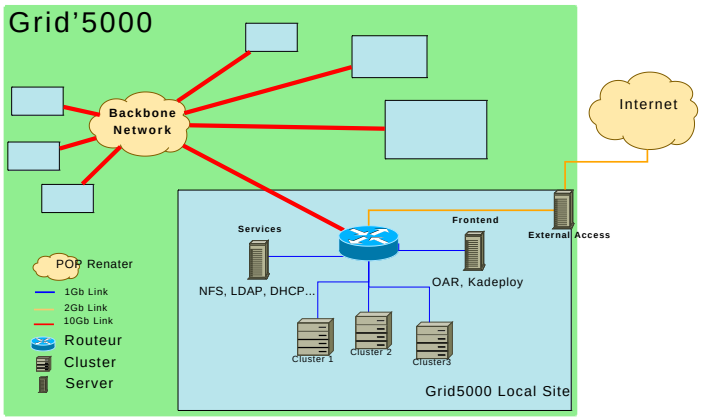

Figure 2. Overview of Grid5000 architecture

life experimentation upon all layers of software stack.

In the remainder of this paper on section 2 we present the Design Concepts and Architectural choices of Grid5000. The various key components of Grid5000 are presented respectively on section 3,4 and 5. Section 3 analyzes OAR, the Resource and Job Management System, section 4 describes Kadeploy reconfiguration mechanism and section 5 presents the various monitoring and experiment steering tools Finally, section 6 gives some conclusions.

2 Grid5000 Design and Architecture

During the preparation of the project in 2003, designers of Grid5000 conducted an analysis on the need of a computer science Grid and the diversity of potential experiments. As described thoroughly on [7], the analysis concluded the need for a large scale (several thousands of CPUs), distributed (10 sites) computer science Grid. Moreover, the experimentation should cover all layers of the software stack, from the application layer to the networking protocols. As the matter of fact, researchers may need a specific experiment setting, different from the other researchers. Researchers involved in networking protocols, Operating Systems and Grid middleware often require a specific OS or kernel for their experiments. Their needs may be quite diverse in Grid Middleware: some require Globus, while others need Unicore, Desktop Grid or P2P middleware. As a consequence, Grid5000 should provide a deep reconfiguration mechanism allowing researchers to deploy, install, boot and run their specific software environments, possibly including all the layers of the software stack. This reconfiguration capability led to the experiment workflow followed by Grid5000 users: 1)reserve specific resources of Grid5000, 2)deploy a software environment on the reserved nodes, 3)run the experiment and make precise measurements, and finally 4)collect results and relieve the machines.

Since researchers are able to boot and run their specific software on Grid5000 sites and machines, the need of security mechanisms arose. Hence we decided to isolate Grid5000 from the rest of the Internet but to let packets fly inside Grid5000 without limitation. The first choice guarantees that Grid5000 will resist to Internet hacker attacks and the second one makes sure that communication performance will not suffer from the overhead of an imposed security system. Thus, Grid5000 is built as a large scale confined cluster of clusters. Strong authentication and authorization checks are done when users log in Grid5000. Grid5000 is composed of 1/3 heterogeneous and 2/3 homogeneous resources, in order to facilitate all kind of experiments.

As analyzed on [8] by Feitelson, the capability to reproduce experimental conditions is fundamental in computer science, especially when performance comparisons are conducted. To fulfill this strong requirement, we decided to use dedicated network links between sites, allow users to allocate dedicated nodes for their experiments and let them install and run their proper experimental condition injectors and measurements software. Thus every user has full control of the allocated Grid5000 partition.

Grid5000 initial goal was to provide 5000 CPUcores distributed over 9 sites in France. Figure 1 shows a map of the initial Grid5000 nation wide grid, while figure 2 presents an overview of the platform's architecture. Every site hosts a cluster and all sites are connected between each other by high speed network links (RENATER 4: 10 Gbps links).

Every user has a single account on Grid5000. Every Grid5000 site manages its own user accounts and runs

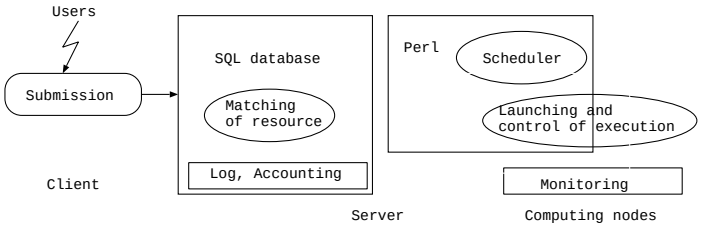

Figure 3. OAR architecture

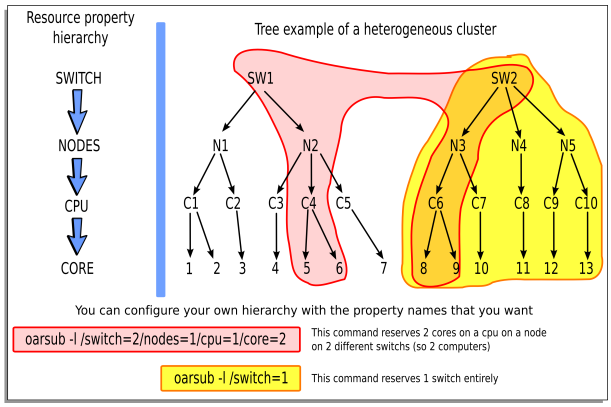

Figure 4. OAR example of usage

an LDAP server. On a given site, the local administrator can manage its user accounts. Once the account is created, the user can access any of the Grid5000 sites or services (monitoring tools, wiki, deployment, etc.). User data are kept local to every site and distribution to remote sites is done by the user through classical file transfer tools (rsync, scp, sftp, etc.). Data transfers from and to the outside of Grid5000 are restricted to secure tools and done through gateway servers.

At cluster level, users submit their resource reservations and experiment jobs using the OAR resource and job management system. To reconfigure the software stack on every reserved node, the users run the Kadeploy toolkit deploying the user defined software environment on a disk partition of selected nodes. Finally to monitor and control the experiments various software tools have been developed by Grid5000 scientists to facilitate different tasks of the experimentation process. In the following sections we present a detailed analysis of these key components used on Grid5000.

3 Cluster resource management and job scheduling: OAR

OAR [9, 10] is an open source Resource Management System for large clusters. Initially developed as a tool for research upon the area of Resource Management and Batch Scheduling, this software has evolved towards a certain 'versatility'. It provides a robust solution, used as a production system in various platforms like the regional grid infrastructure Ciment [1]) used for scientific computations in disciplines like environment, chemistry, astrophysics, etc.

OAR is the Resource and Job Management System selected to function on all Grid5000 sites. It is responsible for resources allocation and reservation along with the jobs execution.

OAR has been designed considering a modular approach with open architectural choices (figure 3). It is based upon high level components: an SQL Database and the Perl/Ruby Script Programming Languages. It can be easily extensible to integrate new features and adapt itself upon different environments.

All the most important functionalities, that exist on commercial Resource Management Systems (like PBSPro or LSF), such as priorities on jobs, advance reservations, resources matching and backfilling are implemented. The priorities are managed through submission queues. All the jobs are submitted to a particular queue which has its own admission rules, scheduling policy and priority. Reservations are a special case in which the user asks for a specific time slot. In this case, as long as the job meet the admission rules and the ressources are available during the requested time slot, the schedule date of the job is definitively set. In OAR, resources required by jobs are matched with available ones. This matching is based on a hierarchical affinity of resources which gives the possibility to allocate from a whole cluster until a specific CPUcore. Moreover a user might need nodes with special properties (like single switch interconnection, or a mandatory quantity of RAM). OAR provides commands to facilitate the allocation of resources. Figure 4 shows an example of OAR jobs submission commands. OAR scheduler also performs backfilling which is the use of idle time slots when large parallel jobs are waiting for execution. Furthermore it can handle *Best Effort* jobs which are low-priority jobs (usually used for desktop grid experiments) that can be cancelled by normal jobs before the end of their allowed execution time.

OAR relies on a specialized parallel launching tool named Taktuk [11, 12] to manage all large-scale operations like parallel tasks launching, nodes probing or monitoring.

In order to be able to execute experiments on different clusters of Grid5000, simultaneously, OARGRID [13] software has been developed which is a wrapper that enables the use of several OAR clusters for easier grid experimentation. It provides the possibility

[1] $https://ciment.ujf-grenoble.fr/cigri$

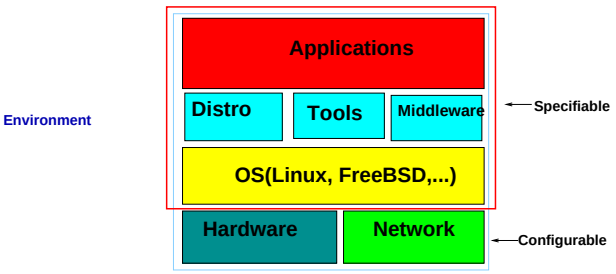

Figure 5. Software environment

of resources allocation and reservation on a grid level. The individual job execution is effectuated by the local OAR system.

4 Reconfiguration mechanism: Kadeploy

In the context of high performance computing research, scientists seem to need various software environments in order to perform their experiments. A software environment contains all the software layers like the operating system, the libraries, the middlewares and the applications (figure 5). According to their experiments nature and the software layer they are investigating (protocols, OS, ..), they often require specific OS. Hence, a tool with a deep reconfiguration mechanism allowing researchers to deploy, install, boot and run their specific software images, is needed.

Kadeploy [14, 15] is a software environment deployment tool designed to solve the above issues providing automated software installation and reconfiguration mechanisms on all the layers of the software stack. Using kadeploy, in a typical experiment sequence, a researcher reserves a partition of the cluster or grid, deploys its software image (figure 7), reboots all the machines of the partition, runs the experiment, collects results and finally relieves the machines. This reconfiguration capability allows researchers to run their experiments in the software environment that perfectly matches their needs and provides to users, a software homogeneous grid.

This tool uses the traditional protocols for network booting: PXE, TFTP, DHCP. As we can see on figure 6, architecture of Kadeploy is designed around a database and a set of specialized operating components. The database is used to store all necessary information for the deployment process, the computing nodes and the environments. At the same time, the code is written in Perl, which is perfectly suited for system tasks. In addition uses a fast mechanism of environment diffusion which depends slightly on the number of nodes. This mechanism is based on a pipeline

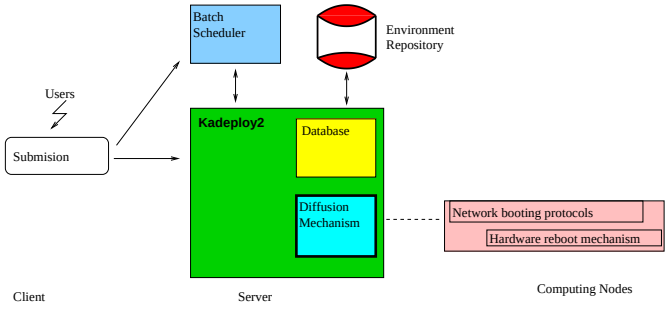

Figure 6. Kadeploy architecture

approach (chain of TCP connections between nodes). This enables operations of deployment on large clusters (1000 nodes).

The deployment process will write a complete environment, on a partition of the disk of each computing node, which will be followed by a reboot on this partition. The process ensures that the partition of the disk where the reference environment of the node is installed, remains intact during diffusion. To guarantee a greater function reliability, Kadeploy tool directs clusters to be coupled with remote mechanisms of hardware reboot. Thus, if a particular problem occurs on one or more nodes during a deployment, a restarting on the reference partition is ordered automatically, on defected nodes.

An environment is created very simply by making an archive of the root partition in compressed `tar` format. To ensure a high level of portability and to permit that an environment is usable on various clusters of similar processor architectures , the environment should not contain information corresponding to the initial cluster. That is possible because the majority of the services have autoconfiguration mechanism (ex: protocol DHCP for the network) and the majority of the operating systems have hardware autodetection mechanisms making it possible to adapt to the minor differences (network cards, disks...). For the services that lack autoconfiguration procedure during the deployment, a procedure known as post-installation process supplements the parameter setting.

Grid5000 uses kadeploy environment deployment tool for effective reconfiguration capabilities.

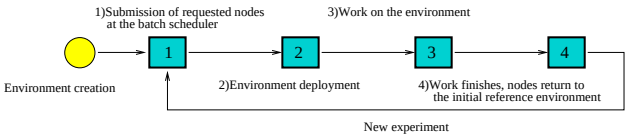

Figure 7. Typical sequence of an environment deployment.

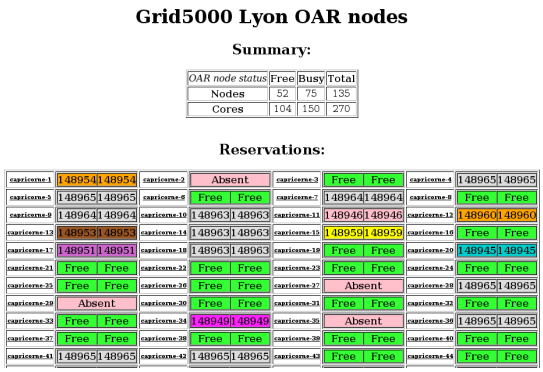

Figure 8. Monika

5 Monitoring and experiment steering tools

In order to control and monitor the experiments various software tools have been developed and used upon Grid5000 platform.

Concerning the initiation and experiment steering tools the following tools are some of the most important software used in the platform: Katapult [16] provides a wrapper to kadeploy for automatic redeployment of nodes in case of failures. Grudu [17] software provides a tool for managing Grid5000 resources, reservations and deployments through a user-friendly web interface. Taktuk [11, 12] software is a tool for deploying parallel remote executions of commands to a large set of remote nodes. KAAPI [18] provides a C++ library that allows to execute multithreaded computation with data flow synchronization between threads.

The experiments and resources usage monitoring is a valuable component of an experimental grid platform. Numerous tools are used to provide detailed information along and after the experimental process. Monika (figure 8) and Gantt (which are parts of OAR software) provide analytical information about the state of jobs and resources of independent clusters or even the whole grid.

Green-net framework [19] is composed by power aware software tools for high performance data transport and computing in large scale distributed systems and it provides ways (web or command line interfaces) to control and measure the energy consumption during the experiments. The electric consumption of some monitored nodes is available live on line, with some graphs, as shown in figure 9.

Finally, Ganglia is used for a finer grain monitoring and kaspied tool provides a set of statistical values about the platform usage.

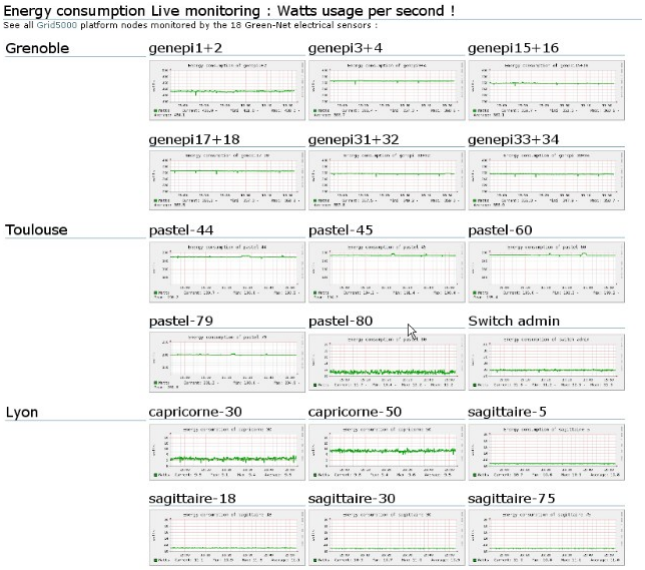

Figure 9. Electrical consumption Monitoring

6 Conclusion

In this article we presented Grid5000 experimental grid platform dedicated for research in computer science. Building a platform like Grid5000 is a full-fledged research topic. The construction of such large scale instruments is new in computer science and the community just starts to get used to deal with all the administrative, technical and scientific details related to the design, construction, exploitation and maintenance of such platforms. Other branches of science like Physics, Astrophysics and Biology have a long history of instrument construction behind them. This is a precious source of inspiration for computer scientists. Apart being instrument to study distributed computing research problems, Grid5000 offers resources opened and shared by a large community of users. Computer scientists find not only a sophisticated environment involving supporting engineers, specific software and dedicated hardware to ease their experiments but also a social context in which they can share their problems, questions and solutions.

Furthermore a lot of international collaborations have been initiated since the beginning of the project, which imply a further opening of the platform in an international scale. As a matter of fact, since July 2009 Grid5000 acquired a new site, in the pool of each resources, situated in the university UFRGS of Porto Alegre in Brazil. This collaboration, gave a new dimension of Grid5000 platform which is now leaning towards an international computer science grid.

References

[1] Cappello, F., Desprez, F., Dayde, M., Jeannot, E., Jegou, 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: Grid2005 6th IEEE/ACM International Workshop on Grid Computing. (2005)

[2] Grid5000: Experimental grid platform. (`http://grid5000.fr`)

[3] PlanetLAB: (`http://www.planet-lab.org/`)

[4] emulab: (http://www.emulab.net/)

[5] GENI: Global environment for network innovations. (`http://www.geni.net/`)

[6] DAS3: (http://www.cs.vu.nl/das3/)

[7] Cappello, F., Bal, H.: Toward an international "computer science grid". Cluster Computing and the Grid, IEEE International Symposium on **0** (2007) 3–12

[8] Feitelson, D.G.: (Experimental computer science: The need for a cultural change)

[9] Capit, N., Costa, G.D., Georgiou, Y., Huard, G., Martin, C., Mounié, G., Neyron, P., Richard, O.: A batch scheduler with high level components. In: 5th Int. Symposium on Cluster Computing and the Grid, Cardiff, UK, IEEE (2005) 776–783

[10] OAR: Resource and job management system. (`http://oar.imag.fr/`)

[11] Claudel, B., Huard, G., Richard, O.: Taktuk, adaptive deployment of remote executions. In: HPDC '09: Proceedings of the 18th ACM international symposium on High performance distributed computing, New York, NY, USA, ACM (2009) 91–100

[12] Taktuk: Adaptive execution deployment. (`http://taktuk.gforge.inria.fr/`)

[13] OARGRID: Oar wrapper for grid level allocation. (`http://gforge.inria.fr/projects/oargrid/`)

[14] Georgiou, Y., Leduc, J., Videau, B., Peyrard, J., Richard, O.: A tool for environment deployment in clusters and light grids. In: Second Workshop on System Management Tools for Large-Scale Parallel Systems (SMTPS'06), Rhodes Island, Greece (2006)

[15] kadeploy: Environment deployment tool. (`http://gforge.inria.fr/projects/kadeploy`)

[16] katapult: automatic experiment deployment. (`http://www.loria.fr/~lnussbau/katapult.html`)

[17] GRUDU: Grid5000 reservation utility for deployment usage. (`http://graal.ens-lyon.fr/DIET/grudu.html`)

[18] KAAPI: (library for ditributed computing)

[19] Green-Net: Power aware framework. (`http://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET/`)