GRID'5000: A LARGE SCALE AND HIGHLY RECONFIGURABLE EXPERIMENTAL GRID TESTBED

Raphaël Bolze[1]
Franck Cappello[2]
Eddy Caron[1]
Michel Daydé[3]
Frédéric Desprez[1]
Emmanuel Jeannot[4]
Yvon Jégou[5]
Stephane Lanteri[6]
Julien Leduc[2]
Noredine Melab[7]
Guillaume Mornet[5]
Raymond Namyst[8]
Pascale Primet[1]
Benjamin Quetier[2]
Olivier Richard[9]
El-Ghazali Talbi[7]
Iréa Touche[10]

Abstract

Large scale distributed systems such as Grids are difficult to study from theoretical models and simulators only. Most Grids deployed at large scale are production platforms that are inappropriate research tools because of their limited reconfiguration, control and monitoring capabilities. In this paper, we present Grid'5000, a 5000 CPU nation-wide infrastructure for research in Grid computing. Grid'5000 is designed to provide a scientific tool for computer scientists similar to the large-scale instruments used by physicists, astronomers, and biologists. We describe the motivations, design considerations, architecture, control, and monitoring infrastructure of this experimental platform. We present configuration examples and performance results for the reconfiguration subsystem.

Key words: Grid, P2P, experimental platform, highly reconfigurable system

The International Journal of High Performance Computing Applications,
Volume 20, No. 3, Fall 2006, pp. 481–494
DOI: 10.1177/1094342006070078

Figures 1–3, 6 appear in color online: http://hpc.sagepub.com

1 Introduction

Grid is well established as a research domain and proposes technologies that are mature enough to be used for real-life applications. Projects such as e-Science (http://www.nesc.ac.uk), TeraGrid (http://www.teragrid.org), Grid3 (http://www.ivdlg.org/grid2003), DEISA (http://www.deisa.org), and NAREGI (http://www.naregi.org/index_e.html/), demonstrate that large-scale infrastructures can be deployed to provide scientists with fairly easy access to geographically distributed resources belonging to different administration domains. Despite its establishment as a workable computing infrastructure, there are still many issues to be solved and mechanisms needed to optimize in performance, fault tolerance, QoS, security, and fairness.

As large-scale distributed systems, Grid software and architecture combine several characteristics which make them difficult to study by following a theoretical approach. Most of the research conducted in Grids is currently performed using simulators, emulators or production platforms. As discussed in the next section, all these tools have limitations making the study of new software and optimizations difficult. Given the complexity of Grids, there is a strong need for highly configurable real-life experimental platforms that can be controlled and monitored directly. Such tools already exist in other contexts. The closest example is PlanetLab (Chun et al. 2003). It consists of a set of PCs connected to the Internet and forming an experimental distributed system. PlanetLab is used for network studies as well as for distributed systems research.

In this paper we present the Grid'5000 http://www.grid5000.org project, still under construction but already in use in France. We first explain the motivation for developing a large scale, real-life experimental platform by discussing the limitations of existing tools. In Section 3, we present the design principles of Grid'5000 which were based on the results of Grid researchers' interviews.

[1]LIP, ENS LYON
[2]INRIA, LRI, PARIS (FCI@LRI.FR)
[3]INPT/IRIT, TOULOUSE
[4]LORIA, INRIA
[5]IRISA, INRIA
[6]INRIA SOPHIA ANTIPOLIS
[7]LIFL, UNIVERSITÉ DE LILLE
[8]LABRI, UNIVERSITÉ DE BORDEAUX
[9]LABORATOIRE ID-IMAG
[10]LGC, TOULOUSE

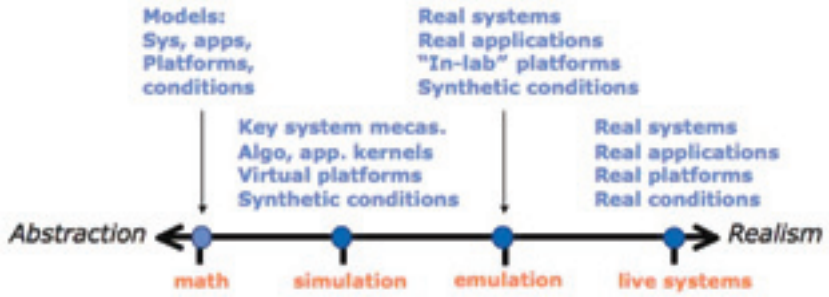

Fig. 1 Methodologies used in distributed system studies.

The implementation of Grid'5000 is described in Section 4. In Section 5 we present evaluation results for the deployment and reboot system, a key component of Grid'5000. Section 6 gives some configuration examples, demonstrating the high reconfigurability of the platform.

2 Motivations and Related Work

As with other scientific domains, research in Grid computing is based on a variety of methodologies and tools. Figure 1 presents the spectrum of methodologies used by researchers to study research issues in distributed systems. In large distributed systems, numerous parameters must be considered and complex interactions between resources make analytical modeling impractical. Thus simulators, emulators, and real platforms are preferred.

Simulators focus on a specific behavior or mechanism of the distributed system and abstract the rest of the system. Their fundamental advantage is their independence of the execution platform. For example, Bricks (Takefusa et al. 1999) was proposed for studies and comparisons of scheduling algorithms and frameworks. Researchers can specify network topologies, server architectures, communication models and scheduling framework components to study multi-client, multi-server Grid scenarios. Some Bricks components are replaceable by real software, allowing validation of external sofware. SimGrid (Casanova, Legrand, and Marchal 2003) is used to study single-client multi-server scheduling in the context of complex, distributed, dynamic, and heterogeneous environments. SimGrid is based on event-driven simulation, providing a set of abstractions and functionalities to build a simulator corresponding to the applications and infrastructures. Resources latency and service rate may be set as constants or evolve according to traces. The topology is fully configurable. GangSim (Dumitrescu and Foster 2005) considers a context where hundreds of institutions and thousands of individuals collectively use tens or hundreds of thousands of computers and the associated storage systems. It models usage policies at the site and Virtual Organization (VO) levels and can combine simulated components with instances of a VO Ganglia Monitoring toolkit running on real resources.

Surprisingly, very few studies have provided validation for these simulators. The validation of Bricks was performed by incorporating NWS (Network Weather Service) in Bricks and comparing the NWS results measured on a real Grid with the ones obtained on a Grid simulated by Bricks. SimGrid validation consisted in comparing the simulator results with the ones obtained analytically on a mathematically tractable problem.

In some situations, complex behaviors and interactions of the distributed system nodes cannot be simulated, because of the difficulty of capturing and extracting the factors influencing the distributed systems. Emulators can address this limitation by executing the actual software part of the distributed system, in its whole complexity. Emulators are generally run on rather ideal infrastructures (i.e. controlled clusters). MicroGrid (Liu, Xia, and Chien 2004) allows researchers to run Grid applications on virtual Grid resources. Resource virtualization is done by intercepting all direct use of resources. The emulation coordination essentially controls the simulation rate, which is determined by the virtualization ratio for all resources. The emulation time base is controlled by a virtualization library returning adjusted times to the system routines. Accurate processor virtualization relies on specific schedulers and the network virtualization (Liu, Xia, and Chien 2004) uses the MaSSF system for a scalable online network simulation.

The authors of MicroGrid have conducted a thorough validation (Liu, Xia, and Chien 2004). The internal timing of MicroGrid was validated using the AutoPilot system. The capacity of the emulator to enforce memory limitation and to maintain the processing model under CPU and I/O competition was validated using microbenchmark. Emulation results were compared with experimental ones on real platforms for the NAS benchmark, in order to validate the full emulation engine. Validation with real applications compared the execution times of CACTUS problem solving environment, Jacobi, ScaLAPACK, Fish, Game of life, and Fasta on real platforms with the ones obtained by MicroGrid. Emulab (White et al. 2002) is another emulator, originally designed for network emulation. It provides advanced controlling mechanisms for the user, allowing the rebooting of nodes in specific OS configurations and the control of the network topology.

Because emulators use the real software, they cannot scale as well as simulators. Furthermore, there is still a gap between emulators and the reality: even traffic and fault injection techniques, generally based on traces or synthetic generators cannot capture all the dynamic, variety and complexity of real-life conditions. Real-life experimental platforms solve this problem by running the real software on realistic hardware. DAS2 (http://www.cs.vu.nl/das2/) is basically an idealized Grid, all sites being connected on the Internet. Experiments are run on top of a Grid middleware managing the classical security and runtime interface issues related to Grid platforms. The nodes are voluntarily homogeneous, providing a much simpler management and helping a better environment for performance comparison (speed up of parallel applications) and understanding. PlanetLab (Chun et al. 2003) is another real-life experimental platform, connecting real machines through the Internet, at the planet scale. Some production Grids (TeraGrid, eScience, DataGrid) have also been used as experimental platforms, before being opened to actual users or during dedicated time slots.

Two major limitations of real-life platforms as experimentation tools are 1) their low software reconfiguration capability and 2) the lack of deep control and monitoring mechanisms for the users. The next section highlights how Grid'5000 addresses these limitations.

3 Designing Grid'5000

The design of Grid'5000 derives from the combination of 1) the limitations observed in simulators, emulators and real platforms and 2) an investigation into the research topics conducted by the Grid community. These two elements led to the proposal for a large scale experimental tool, with deep reconfiguration capability, a controlled level of heterogeneity and a strong control and monitoring infrastructure.

3.1 Experiment Diversity

During the preparation of the project (2003), we asked researchers in Grid computing which experiments they were willing to conduct on a large scale real-life experimental platform. The members of 10 teams in France, involved in different aspects of Grid computing and well connected to the international Grid community, proposed a set of about 100 experiments. It was surprising to discover that almost all teams required different infrastructure settings for their experiments. The experiment diversity nearly covered all layers of the software stack used in Grid computing: networking protocols (improving point to point and multipoints protocols in the Grid context, etc.); operating systems mechanisms (virtual machines, single system image, etc.); Grid middleware; application runtimes (object oriented, desktop oriented, etc.); applications (life science, physics, engineering, etc.); problem solving environments. Research in these layers concerns scalability (up to thousands of CPUs), performance, fault tolerance, QoS, and security.

3.2 Deep Reconfiguration

For researchers involved in network protocols, OS and Grid middleware research, the software setting for their experiments often requires specific OS. Some researchers need Linux, while others are interested in Solaris10 or Windows. For networking research, FreeBSD is preferred because network emulators such as Dummynet and Modelnet run only on this operating system. Some researchers also need to test and improve protocol performance (for example changing the size of the TCP window or testing alternative protocols). Some research on virtual machines, process checkpointing and migration need the installation of specific OS versions or OS patches that may not be compatible with each other. Even for experiments over the OS layers, researchers have some preferences: for example some prefer Linux kernel 2.4 or 2.6 because their schedulers differ. Researchers' needs are quite different in Grid middleware: some require Globus (in different versions: 3.2, 4, DataGrid version) while others need Unicore, Desktop Grid or P2P middleware. Some other researchers need to make experiments without any Grid middleware and test applications and mechanisms in a multi-site, multi-cluster environment before evaluating the Middleware overhead. According to this inquiry on researchers' needs, Grid'5000 should provide a deep reconfiguration mechanism allowing researchers to deploy, install, boot and run their specific software images, possibly including all the layers of the software stack. In a typical experiment sequence, a researcher reserves a partition of Grid'5000, deploys its software image, reboots all the machines of the partition, runs the experiment, collects results and relieves

the machines. This reconfiguration capability allows all researchers to run their experiments in the software environment exactly corresponding to their needs.

3.3 A Two-Level Security Approach

Because researchers must be able to boot and run their specific software stack on Grid'5000 sites and machines, we cannot make any assumption on the correct configuration of the security mechanisms. As a consequence, we should consider that Grid'5000 machines are not protected. Two other constraints increase the security issue complexity: 1) all the sites hosting the machines are connected through the Internet and 2) basically inter-site communication should not suffer any platform security restriction and overhead during experiments. From this set of constraints, we decided to use a two-level security design with the following rules: a) Grid'5000 sites are not directly connected to the Internet and b) all communication packets fly without limitation between Grid'5000 sites. The first rule ensures that Grid'5000 will resist hacker attacks and will not be used as basis of attacks (i.e. massive DoS or other more restricted attacks).

These design rules led to building a large scale confined cluster of clusters. Users connect to Grid'5000 from the lab where the machines are hosted. Rigorous authentication and authorization check is done first to enter the lab and then to log in Grid'5000 nodes from the lab. In order to participate in multiplatform experiments, it is possible for Grid'5000 sites to open restricted routes through the Internet to external clusters (called satellite sites).

3.4 Two Thirds as Homogeneous Nodes

Performance evaluation in Grid is a complex issue. Speedup evaluation is hard to evaluate with heterogeneous hardware. In addition, the hardware diversity increases the complexity of the deployment, reboot and control subsystem. Moreover, multiplying the hardware configurations directly leads to an increase in the everyday management and maintenance cost. Considering these three parameters, we decided that 2/3 of the total machines should be homogeneous. However, Grid are heterogeneous by nature and this is an important dimension in the experiment diversity. This is the reason why we chose to keep 1/3 as heterogeneous machines.

3.5 Precise Control and Measurement

Grid'5000 is used for Grid software evaluation and making fair comparisons of alternative algorithms, software, protocols, etc. This implies two elements: first, users should be able to steer their experiments in a reproducible way and second, they should be able to access probes providing precise measurements during the experiments. The reproducibility of experiment steering includes the capability to 1) reserve the same set of nodes, 2) deploy and run the same piece of software on the same nodes, 3) synchronize the experiment execution on all the involved machines, 4) if needed, repeat sequences of operations in a timely and synchronous way, 5) inject the same experimental conditions (synthetic or trace based: fault injection, packet loss, latency increase, bandwidth reduction). As described in the next section, Grid'5000 software set provides a reservation tool (OAR, see Georgiou et al. 2005), a deployment tool (Kadeploy2, see Georgiou et al. 2006) and several experimental condition injectors. Precise and extensive measurement is a fundamental aspect of experimental evaluation on real-life platforms.

Global observation of the network (from its edges) and local observation of processor, memory or disk is difficult at the hardware level and since the users may use their own software configuration, there is no way to provide a built-in and trustworthy monitoring system for CPU, memory and disc. Hence, it is the responsibility of the users to properly install, configure and manage the software observation tools they need for their experiments.

4 Grid'5000 Architecture

The Grid'5000 architecture implements the principles described in the previous section. Based on the researchers requirements, the scalability needs and the number of researchers, we decided to build a platform of 5000 CPUs distributed over 9 sites in France. Figure 2 presents

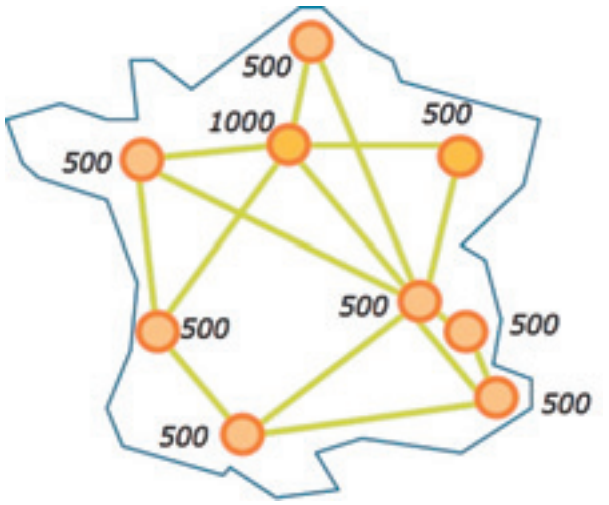

Fig. 2 Overview of Grid'5000.

an overview of Grid'5000. Every site hosts a cluster and all sites are connected by high speed network (a novel network architecture is being deployed, connecting the sites with 10 Gbps links).

Numbers in Figure 2 give the target number of CPUs for every cluster. Two-thirds of the nodes are dual CPU 1U racks equipped with 2 AMD Opteron processors running at 2 GHz, 2 GB of memory and two 1Gbps Ethernet Adapters. Clusters are also equipped with high speed networks (Myrinet, Infiniband, etc.). In the rest of this section we present the key architectural elements of Grid'5000.

4.1 A Confined System

As discussed earlier, the Grid'5000 architecture should provide an isolated domain where communication is allowed without restriction between sites and is not possible directly with the outside world. Mechanisms based on state-of-the-art technology such as public key infrastructures and X509 certificates, produced by the Grid community to secure all resources accessed are not suitable for the Grid'5000. The GSI high level security approach imposes a heavy overhead and impacts on the performance, biasing the results of studies not directly related to security.

A private dedicated network (PN) or a virtual private network (VPN) are, then, the only solutions to compose a secure grid backbone and to build such a confined infrastructure. In Grid'5000, we chose to interconnect the sites with a combination of DiffServ and MPLS technology (Multiprotocol label switching) provided by RENATER (our service provider). MPLS is an efficient way to build secure virtual private networks. As the packet encapsulation is done at very low level by very high performance routers, the overhead is negligible and has no impact on the end-to-end performance. One difference between classical Internet and MPLS, is that MPLS fixes the routing of Grid'5000 datagrams and flows. We consider that the static routing constraint is reasonable for such a testbed. Concerning the background traffic, and the meaningfulness of our end-to-end measures, we chose to let the researcher load the network with artificially generated traffic he can monitor rather than letting him deal with unknown Internet traffic. It allows the calibration of tools, the debugging of protocols and the investigation of alternative traffic control strategies and different types of traffic models. It appears Grid'5000 is complementary to PlanetLab as our instrument enables fine tuning and good understanding of basic phenomenon in the absence of extra noise. PlanetLab offers a more realistic view of the present Internet behavior, but cannot capture behaviors at the limit of the resource capacities and potentially the future Internet behavior.

Many VPN implementation solutions are available but they do not provide security and QoS guarantees simultaneously. For security, network layer VPNs may use tunneling or network layer encryption (layer 3 VPN). A link layer, VPNs such as MPLS are directly provided by network service providers (layer 2-3 VPN). The advantage of the MPLS VPN over IP VPN (Ipsec) is performance. As Grid'5000 sites are connected to the same NREN (National Research and Education Network), the multi-domain issue of the MPLS technology is avoided here. For performance guarantee, a combination of DiffServ and MPLS will be configured for Grid'5000 links. The Premium service will be used for delay and bandwidth guarantees required for reproducible experimental conditions and performance measurements. This MPLS-based Grid architecture allows the creation of a trust context that even enables to experiment with new security solutions for IP VPN-based Grids. Figure 3 presents the resulting communication architecture.

Using MPLS in Grid architecture is not an isolated choice. Recently, a Grid VPN research group was born within the GGF, attesting a real interest in developing and using MPLS, G-MPLS or lower level optical switching technologies for the Grid.

4.2 User View and Data Management

As previously mentioned, communications are done with minimal authentication between Grid'5000 machines. The logical consequence is that a user has a single account across the whole platform. However, each Grid'5000 site manages its own user accounts. Reliability of the authentication system is also critical. A local network outage should not break the authentication process on other sites. These two requirements have been fulfilled by the installation of an LDAP directory. Every site runs an LDAP server containing the same tree: under a common root, a branch is defined for each site. On a given site, the local administrator has read-write access to the branch and can manage its user accounts. The other branches are periodically synchronized from remote servers and are read-only.

From the user's point of view, this design is transparent. Once the account is created, the user can access any of the Grid'5000 sites or services (monitoring tools, Wiki, deployment, etc.). His data, however, are local to every site. They are shared on any given cluster through NFS, but distribution to another remote site is done by the user through classical file transfer tools (rsync, scp, sftp, etc.). Data transfers with the outside of Grid'5000 are restricted to secure tools to prevent identity spoofing and public key authentication is used to prevent brute-force attacks.

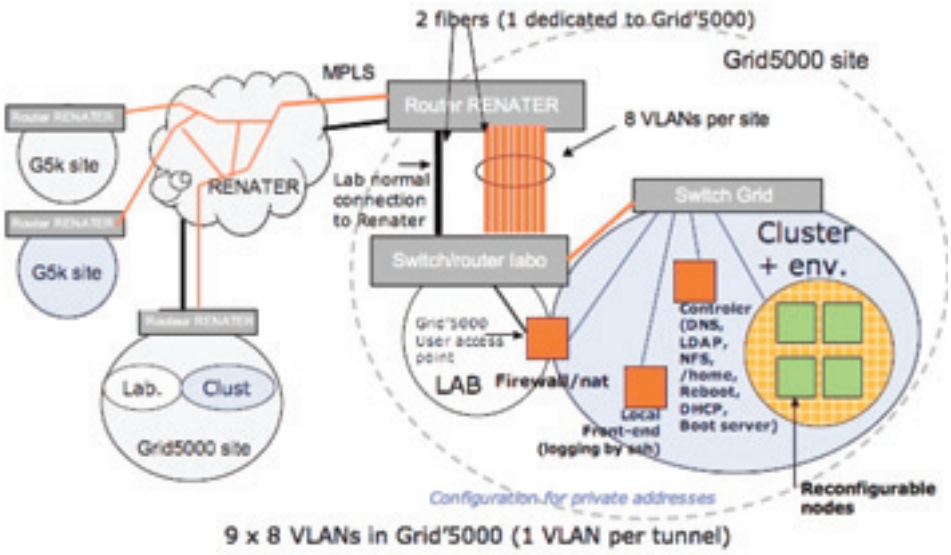

Fig. 3 Communication architecture.

4.3 Experiment Scheduling

Experiment scheduling and resource allocation is managed by a resource management system called OAR (Georgiou et al. 2005) at cluster level and by a simple broker at the grid level. OAR architecture is built from a relational database engine *MySql*. All large-scale operations such as parallel task launching, node probing or monitoring are performed using a specialized parallel launching tool named *Taktuk* (Augerat, Martin, and Stein 2002). OAR provides most of the important features implemented by other batch schedulers such as priority scheduling by queues, advance reservations, backfilling and resource match making.

At grid level, a simple broker allows co-allocating sets of nodes on every selected cluster.

The co-allocation process works as follows: 1) user submits an experiment which needs several sets of nodes on different clusters; 2) in round-robin sequence, the broker submits a reservation to each local batch scheduler. If one reservation is refused, all previously accepted reservations are canceled. When all local reservations are accepted, the user receives an identifier from the broker, allowing the user to retrieve information about the allocated set of nodes.

In Grid'5000, the resource management system is coupled with node reconfiguration operation at different points. First, a specific queue is defined where users can submit experiments requesting node reconfiguration. Second, there is a dynamic control of deployment rights in the prologue script that is executed before starting the experiment. This gives the user the capability of deploying system images on the allocated node partition. Rights are revoked in the epilogue script after the experiment. Third, after the completion of experiments involving node reconfiguration, all nodes are rebooted in a default environment. This default environment provides libraries and middleware for experiments without reconfiguration.

4.4 Node Reconfiguration

Node reconfiguration operation is based on a deployment tool called Kadeploy2 (Georgiou et al. 2006). This tool allows users to deploy their own software environment on a disk partition of selected nodes. As previously mentioned, the software environment contains all software layers from OS to application level needed by users for their experiments.

The architecture of Kadeploy2 is also designed around a database and a set of specialized operating components.

The database is used to manage different aspects of the node configuration (disk partition schemes, environment deployed on every partition), user rights to deploy on nodes, environment description (kernel, initrd, custom kernel parameters, desired filesystem for environment, associated postinstallation) and logging of deployment operations.

Several deployment procedures are available, depending mainly on OS type and filesystem specificity. We only sketch the usual deployment procedure. First, when a user initiates a deploy operation, he provides an environment name allowing the retrieval of associated information from the database. The user provides this information at environment registration. Deployment begins by rebooting all nodes on a minimal system through a network booting sequence. This system prepares the target disk for deployment (disk partitionning, partition formatting and mounting). The next step in the deployment is the environment broadcast which uses a pipelined transfer between nodes with on-the-fly image decompression. At this point, some adjustments must be done on the broadcasted environment in order to be compliant with node and site policies (mounting tables, keys for authentication, information for specific services that cannot support auto-configuration). The last deployment step consists in rebooting the nodes on the deployed system from a network loaded bootloader.

5 Deployment System Evaluation

In this section, we present the evaluation of the deployment and reboot system of Grid'5000. Evaluation of other parts of Grid'5000 will be presented in future papers. The deployment and reboot system is certainly the most important mechanism of Grid'5000, enabling a rapid turnaround of experiments on the platform. Typical deployment and reboot mechanisms for clusters cannot be coupled to a batch scheduler. Moreover, they are not designed to concurrently install different systems on separate cluster partitions. Our objective is to provide a reconfiguration time (boot-to-boot: B2B) lower than 10 minutes for the 5000 CPUs of the platform. This means: 1) deploying the software image on all the nodes of every site (a site may contain up to 500 nodes); 2) issuing the reboot order on all Grid'5000 nodes; and 3) the reboot of all nodes from the deployed software image. As previously mentioned, Kadeploy2 uses more steps, booting a light kernel to prepare the user partition to boot from for the experiment.

The B2B time depends not only on the performance of Kadeploy2 but also on the OS to be booted (OS have different configurations and run different sets of services). Figure 4 presents the B2B time according to the number of nodes, in a single site, for a simple kernel without service, on a cluster of 200 nodes.

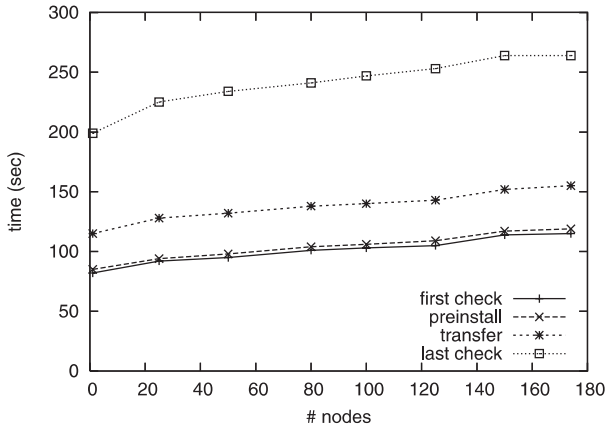

Fig. 4 Time (in seconds) to deploy and boot a new OS on a cluster with Kadeploy2.

The figure presents the completion time of every step included in the B2B time (as a cumulated graph): 1) the time to boot the preparation OS launching a light kernel (first check); 2) the time to prepare the disk partitions before the installation of the user environment (preinstall); 3) the time to transfer the user environment archive (transfer); and 4) the time to boot the user OS (lastcheck). First, the figure shows that the boot time depends on the number of nodes. This is because the boot time is different for all machines and we consider only the slowest one. In contrast, the disk preparation and environment transfer times increase negligibly with the number of nodes. The time to reboot the 2 OS largely dominates the environment transfer time. Altogether, the figure clearly shows a B2B time evolving linearly with the number of nodes following an affine function that could be evaluated as $B2B_{time}$ = 200 secs + (0.33 × X), X being the number of nodes.

Figure 5 presents the time diagram of a deployment and reboot phase involving 2 Grid'5000 sites for a total of 260 nodes (180 nodes in site 1 and 80 nodes in site 2). The vertical axis corresponds to the number of nodes in deployment and reboot states. At t = 0 s, all the nodes are running an OS. At t = 30 s, a deployment sequence is issued. At t = 50 s, all nodes are rebooting the deployment kernel. At t = 160 s all nodes have rebooted and are preparing the user partition. The clusters start the second reboot at t = 200 s for site 2 and t = 340 s for site 1. Site 2 nodes are rebooted with the user OS at t = 320 s. All nodes are rebooted with the user OS (including Site 1) at t = 450 s. At t = 800 s, the user experiment is completed and a reboot order is issued making all nodes reboot to default environment. This figure demonstrates that the current B2B time at the Grid level (450 seconds) is well

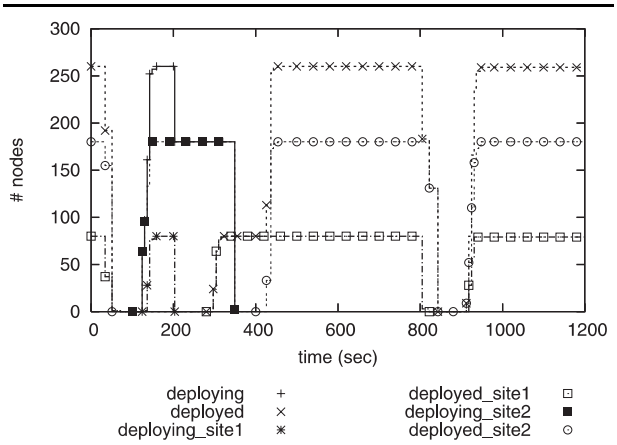

Fig. 5 Time diagram for the deployment and reboot of a user environment on 2 sites.

below the 10 minute mark. The deployment and reboot system is still in Alpha version. It is not tuned and there are many optimization opportunities (Georgiou et al. 2006).

6 Grid'5000 Configuration Examples

The main objective of the Grid'5000 set of software is to ease the deployment, execution and result collection of large scale Grid experiments. In this section, we present 5 examples for Grid'5000 reconfiguration for experiments in networking protocols, Grid middleware infrastructures, and GridRPC environment.

6.1 Testing Recent P2P Protocols in Grid Context

BitTorrent is a popular file distribution system outperforming FTP performance when delivering large and highly demanded files. The key idea of BitTorrent is the cooperation of the downloaders of the same file by uploading chunks of the file to each other. As such, BitTorrent is a nice broadcast protocol for large files in data and computational Grids. BitTorrent uses TCP as the transport protocol.

In this section, we describe how we can deploy, run and collect experiment results, when performing simple BitTorrent performance evaluation for a variation of TCP protocol, on homogeneous nodes of Grid'5000. The modification of the TCP stack involves the compilation and deployment of a specific OS kernel. The experiment requires 9 steps: Step 1) BitTorrent code is instrumented to log reception and emission events (type of communication, sender identifier, receiver identifier, time and chunk identifier). BitTorrent has been instrumented to replay the logged sequence of events. Step 2) The software image is prepared (installing specific libraries and software – Python for BitTorrent), based on a minimal image certified to work on the experimental nodes. The kernel is patched and compiled with alternative TCP versions. The local root file system is then archived and registered on the deploying software database on all sites. Step 3) Nodes are reserved possibly from the same selection file, using OAR. Step 4) The archived file system image is deployed on a user-specified partition of all nodes, using Kadeploy2. Step 5) Kadeploy2 reboots all the reserved nodes and checks that the machine is responding to ping and ssh. 6) The BitTorrent file to be broadcasted, is stored on the user home directory where the BitTorrent master node (the seeder) will run. The list of nodes provided by OAR is stored on the BitTorrent master node. 7) Node clocks are synchronized using NTP-date. 8) A distributed launcher program controls the start of the experiment script on all the nodes. The BitTorrent tracker is started first, then the Torrent file created is registered in the tracker, then the seeder is started on the master and finally, the clients (leechers) are started on all the other nodes. The BitTorrent events are recorded locally on all the nodes. 9) All log files are collected and stored in the user home directory of the user site gateway. Reserved nodes are released.

6.2 Deploying a Globus Toolkit

Globus is an open source grid middleware toolkit used for building grid systems and applications. This part describes how we can map a Globus (Toolkit 2) virtual grid on Grid'5000, deploy Globus, and run experiments. The topology we chose for our virtual Globus grid was to have one Globus installation on each Grid'5000 site. We consider each site to be a separate cluster that provides services through the Globus Toolkit. Since we are emulating a grid, each cluster manages its own user accounts (i.e. no grid-wide user directory). Job execution on clusters is managed by a batch job scheduler (e.g. OAR, PBS). Each cluster manages user accounts and job scheduling with their software of choice, as we only need homogeneity inside clusters. Each site runs a certification authority (CA) that delivers user certificates for their users, as well as host certificates. We pick a front node on each site, and install Globus services on this front node. These services accept requests from other sites, authenticate and authorize them, then perform an action (e.g. submit a job) on behalf of the client. Clients authenticate services with a host certificate delivered by the site the services run on. The Gatekeeper maps user certificates to the user accounts of each cluster, and executes them with the local job scheduler. Front nodes also run the MDS (monitoring and discovery system) service, and GSIFTP (data transfer).

Globus toolkit is deployed by creating a system image that contains a Globus installation tailored for the experiment (since we deploy the whole system image, everything can be customized up to the operating system kernel). We create for each site an image for cluster compute nodes with a batch scheduler, and an image for the front node with the Globus Toolkit services (Gatekeeper, MDS, GSIFTP, and certificates). The virtual Globus grid is deployed on Grid'5000 machines using the Kadeploy tools, thereby turning Grid'5000 into a virtual Globus grid as long as the Kadeploy reservation lasts. While Globus users are running their experiments, log files are saved to the local drives of each node. As soon as the experiment is done, Kadeploy reboots the nodes with their default system image, and users can retrieve their log files and process them.

6.3 A Corba Based Grid Running DIET and TLSE

The DIET (Caron and Desprez 2006) middleware infrastructure follows the GridRPC paradigm (Seymour et al. 2004) for client–server computing over the Grid. It is designed as a set of hierarchical components (client, master and local agents, and server daemons). It finds an appropriate server according to the information provided in the client request (problem to be solved, size of the data involved), the performance of the target platform (server load, available memory, communication performance), and the availability of data stored during previous computations. The scheduler is distributed using several hierarchies connected either statically (in a Corba fashion) or dynamically (in a peer-to-peer fashion).

The main goal of the Grid-TLSE project (Dayde et al. 2004) is to design an expert site that provides an easy access to a number of sparse matrix solver packages allowing their comparative analysis on user-submitted problems, as well as on matrices from collections also available on the site. The site provides user assistance in choosing the right solver for its problems and appropriate values for the solver parameters. A computational Grid managed by DIET is used to deal with all the runs related to user requests. Our goal in the Grid'5000 project is twofold. First we want to validate the scalability of our distributed scheduling architecture at a large scale (using thousands of servers and clients) and then to test some deployments of the TLSE architecture for future production use.

In the current availability of Grid'5000 platform, the deployment of DIET with TLSE server works in three phases. The first step consists in sending one OAR request at each site, to reserve a maximum of available nodes. The second phase consists in receiving OAR information to know which nodes are given by reservation. The third phase generates an XML file with the dynamic information as well as names of nodes at each site. These files will be used by GoDIET to deploy DIET. Our main goal during this first experience is to corroborate a theoretical study of the deployment with the hardware capability of Grid'5000 platform (CPU performance, bandwidth, etc.) to design a hierarchy that achieves a good scalability and a good efficiency for DIET. From this XML file, GoDIET deploys agents (or schedulers), servers and services bound to DIET as Corba services (i.e. naming service) along with a distributed log tool designed for the visualization tools (VizDIET, see Bolze, Caron, and Desprez 2006). Figure 6 shows a large deployment of DIET using 574 computing nodes and 9 agents for the scheduling of 45000 requests. The 574 servers are deployed on 8 clusters and 7 sites.

6.4 Process Design, Optimization, Planning and Scheduling

Process systems engineering is concerned with the understanding and development of systematic procedures for the design and operation of chemical process systems, ranging from continuous to batch processes at industrial scale (Ponish et al. 2005). More precisely, the optimal design continuous process consists in selecting simultaneously the unit operations, the topology and the best operating conditions. Several software tools have been developed for solving this type of problems. One of them, AG (1,2), used at the LGC (Laboratoire de Genie Chimique) is a serial fortran code. To give an illustration, the treated problem instances involve problem sizes of between 170 and 210 variables. Half of them are integer, which corresponds to a combinatorial aspect of about 1.e40 and 1.e50 (since 3 values are possible for each integer variable). The problem is identified as an NP-hard problem.

This application is multi-parametric by nature since it uses a stochastic algorithm where the execution is repeated 100 times with different parameters. Each execution requires 4 hours on the previous example giving a total execution time of 400 hours on a single PC.

Gridification of such applications is straightforward. The first step consists in modifying the code in order to be able to schedule the 100 executions over 100 different nodes. The code has then been deployed over 5 clusters of Grid'5000: Lyon, Orsay, Sophia, Bordeaux, and Toulouse. The code does not reference any external library which simplifies greatly the installation process at every site.

Getting 100 nodes over 5 clusters of Grid'5000 is usually not a problem and the elapsed time for simulation is reduced from 400 hours down to 4 hours. There is an obvious benefit: the possibility of solving larger problems. But a major benefit compared with traditional large computer infrastructures is that the time between execution launching (usually through a batch system that may

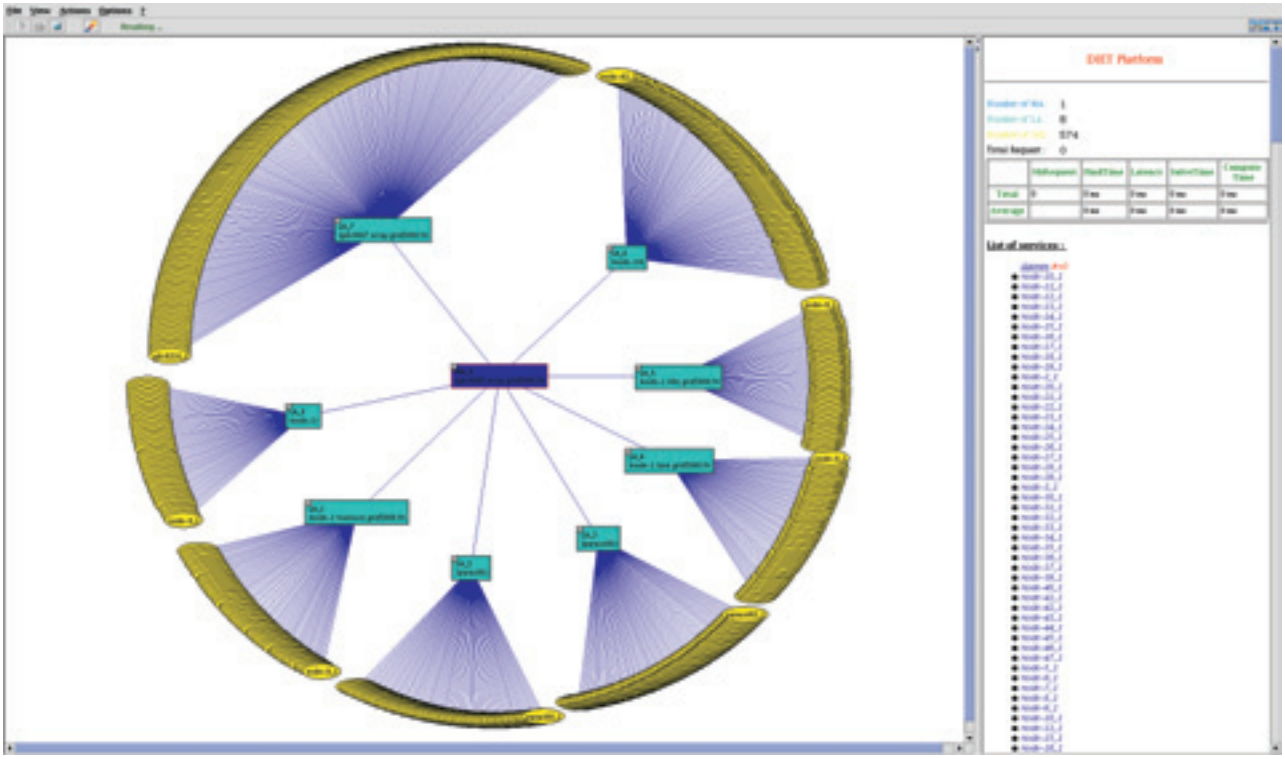

Fig. 6 Large DIET deployment on Grid'5000.

limit the maximal number of processors to a lower number and impose a long wait on a specific queue) and result recovering is drastically reduced, allowing researchers to carry out more simulations.

6.5 The Flow-Shop Challenge on Grid'5000

The Flow-Shop problem consists roughly in finding a schedule of a set of jobs on a set of machines that minimizes the total execution time called the *makespan*. The jobs must be scheduled in the same order on all machines, and each machine cannot be simultaneously assigned to two jobs. The complexity of the problem is very important for large size instances in terms of potential solutions (i.e. schedules). Even with a modern workstation the resolution based on an exhaustive enumeration of all possible combinations would take several years. Therefore, the challenge is to reduce the number of explored solutions using efficient algorithms in order to solve the problem in a reasonable time. Nevertheless, even if these algorithms allow significant reduction in the size of the search space the complexity remains high, and the problem could not be efficiently solved without computational grids.

To solve the problem, a new grid exact method based on the Branch-and-Bound (B&B) algorithm has been proposed by Melab (2005). The method is based on a large scale dispatcher–worker cycle stealing approach. The dispatcher controls the exploration of the search tree generated by the distributed B&B algorithm. It maintains the best solution found so far and a pool of work units and ensures their dynamic allocation to the different workers joining the computational grid. Each work unit represents a set of nodes (or solutions) to be or being explored, and is designated by a small descriptor. Each worker explores its assigned tree nodes using the B&B algorithm and sends back a solution to the dispatcher if it is better than the best known solution so far. To deal with the load bal-

ancing issue, as soon as its local pool of nodes to be explored is empty each worker requests from the dispatcher a work unit. The dispatcher selects a work unit being executed and splits it in two parts. The second part, which is probably not yet explored, is sent to the worker asking for work.

Another issue which is dealt with in the proposed grid exact method is the fault tolerance. This issue arises on a computational grid because of failures of resources (processors, networks, etc.) or their dynamic availability. Within the Grid'5000 context, the dynamic availability is a result of the reservation policy of the grid. Indeed, for long-running applications, a series of reservations is required to resume their execution. The end of each reservation is put in the same category as a failure of the corresponding resources. In the proposed method, a checkpointing-based approach is proposed to deal with the fault-tolerance issue. Each worker periodically requests the dispatcher to update the descriptor of its associated work unit with its current state. The descriptor is based on a special coding of the search tree which allows minimizing of the memory space required by the checkpointing mechanism and the communications involved by the dynamic work distribution.

The proposed method has been implemented using the XtremWeb middleware (Mezmaz, Melab, and Talbi 2006) and RPCs. The second implementation is used to solve the Taillard's Flow-Shop problem instance $Ta056$[1] of scheduling 50 jobs on 20 machines. A near-optimal solution has been found in (Ruiz and Stutzle 2004). However, as it is CPU time consuming such instance has never been optimally solved. The proposed method in Melab (2005) allowed not only an improvement in the best known solution (Ruiz and Stutzle 2004) for the problem instance but also proved the optimality of the provided solution. Indeed, once the supposed optimal solution is found it has to be compared with the remaining solutions being visited to prove that it is really the best.

The experiments were performed on a computational grid including, simultaneously, processors from Grid'5000 and different educational networks of Université de Lille1 (Polytech'Lille, IEEA, IUT "A"). The number of processors averaged approximately 500, and peaked at 1245 machines during one night. The Grid'5000 sites involved in the computation are Bordeaux, Lille, Orsay, Rennes, Sophia-Antipolis and Toulouse. The optimal solution was found with a total wall-clock time of 7 weeks. The experiment was performed a second time starting from the best known near-optimal solution minus 1. The optimal solution was found within 25 days and 46 minutes. The resolution would take 22 years 185 days and 16 hours on a single machine. During the resolution, an average of 328 processors were used and peaked at 1195 processors. The total number of explored nodes was $6{,}5874e+12$, and a small number of them (0. 39%) were explored twice (redundant work). Moreover, 129 958 work allocation operations and 4 094 176 checkpointing opeartions have been performed. Such statistics show that the dynamic load-balancing and checkpointing mechanisms have been heavily requested and performed well. Furthermore, the parallel efficiency is measured as the ratio between the agregated execution time of the workers and the total time of their availability. The parallel efficiency observed in this experience is 97%, so the load balancing approach is very efficient. Finally, the average CPU time consumed by the dispatcher is 1.7%. The approach scales up to 1195 processors without any problem.

7 Conclusion

Grid'5000 belongs to a novel category of research tools for Grid research: a large scale distributed platform that can be easily controlled, reconfigured, and monitored. We have presented the motivation behind design and architecture of this platform. The main difference between Grid'5000 and previous real-life experimental platforms is its degree of reconfigurability, allowing researchers to deploy and install the exact software environment they need for each experiment. This capability raises a security difficulty, solved in Grid'5000 by establishing a virtual domain spanning over several sites, rigorously controlling the communications at the domain boundaries and relaxing restrictions for intra-domain communications. We have described some configuration examples, illustrating the variety of experiments that can benefit from Grid'5000. We also presented the performance of the reconfiguration system which provides a "boot-to-boot" time of less than 10 minutes on the full platform.

Ongoing work focuses on several areas: 1) ease software image construction for the users; 2) provide automatic validation of software images; 3) support and coordinate experiments; and 4) tune and validate network performance.

For more information about the Grid'5000 project, please contact the corresponding author, Franck Cappello (fci@lri.fr), who is responsible for this project.

Acknowledgments

We would like to thank the French Ministry of research and the ACI Grid and ACI Data Mass incentives, especially Thierry Priol (Director of the ACI GRID) and Brigitte Plateau (Head of the Scientific Committee of the ACI Grid), and Dany Vandromme (Director of RENATER) for their support. We also thank INRIA, CNRS, regional councils of Aquitaine, Bretagne, Ile de France and Provence Alpe Côte d'Azur, Alpes Maritimes General Council and the following Universities: University of Paris Sud,

Orsay, University Joseph Fourier, Grenoble, University of Nice-Sophia Antipolis, University of Rennes 1, Institut National Polytechnique de Toulouse/INSA/FERIA/ Universite Paul Sabatier, Toulouse, University Bordeaux 1, University Lille 1/GENOPOLE, Ecole Normale Superieure de Lyon. We also thank MYRICOM.

Author Biographies

Raphaël Bolze is a Ph.D. student at Ecole Normale Superieure de Lyon. He received his Masters in computer science in 2003 from the Institut de Recherche en Informatiques de Nantes and also a Masters degree in engineering from l'Ecole Polytechniques de l'Universite de Nantes. His research interests focus on workflow scheduling over grid environment.

Franck Cappello holds a Research Director position at INRIA and leads the Grand-Large project at INRIA. He has initiated the XtremWeb (Desktop Grid) and MPICH-V (Fault tolerant MPI) projects. He is currently the director of the Grid'5000 project, designing, building and running a large scale Grid experimental platform. He has authored more than 60 papers in the domains of high performance programming, desktop grids, grids and fault tolerant MPI. He has contributed to more than 30 Program Committees. He is editorial board member of the International Journal on GRID Computing and steering committee member of IEEE HPDC and IEEE/ACM CCGRID. He is the general chair of IEEE HPDC'2006.

Eddy Caron is an assistant professor at Ecole Normale Superieure de Lyon and holds a position with the LIP laboratory (ENS Lyon, France). He is a member of GRAAL project and technical manager for the DIET software package. He received his Ph.D. in computer science from University de Picardie Jules Verne in 2000. His research interests include parallel libraries for scientific computing on parallel distributed memory machines, problem solving environments, and grid computing.

Michel J. Daydé received his Ph.D. from Institut National Polytechnique de Toulouse (France) in 1986 in computer science. From 1987 to 1995, he was a postdoctorate fellow then visiting a Senior Scientist in the Parallel Algorithms Group at CERFACS. From 1988, he has been Professor at Ecole Nationale Superieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et des Télécommunications (ENSEEIHT) at Institut National Polytechnique de Toulouse. Since 1996, he has been Research Director in the Groupe Algorithmes Paralleles et Optimisation at Institut de Recherche en Informatique de Toulouse (IRIT). He is Head of the ENSEEIHT Site of IRIT and Vice-Head of IRIT. His current research interests are in grid computing, parallel computing and computational kernels in linear algebra and large scale nonlinear optimization. He is the coordinator of the GRID-TLSE Project and scientific coordinator of the Toulouse/Midi-Pyrenees Site of GRID'5000.

Frédéric Desprez is a director of research at INRIA and holds a position at LIP laboratory (ENS Lyon, France). He received is Ph.D. in computer science from the Institut National Polytechnique de Grenoble in 1994 and his M.S. in computer science from the ENS Lyon in 1990. His research interests include parallel libraries for scientific computing on parallel distributed memory machines, problem solving environments, and grid computing.

Emmanuel Jeannot is currently full-time researcher at INRIA (Institut National de Recherche en Informatique et en Automatique) and is doing its research at the LORIA laboratory. From September 1999 to September 2005 he was associate professor at the Université Henry Poincare, Nancy 1. He got his Ph.D. and Master degree of computer science (respectively in 1996 and 1999) both from Ecole Normale Superieure de Lyon. His main research interests are scheduling for heterogeneous environments and grids, data redistribution, grid computing software, adaptive online compression and programming models. He is currently visiting the ICL Laboratory of the University of Tennessee.

Yvon Jégou is a full time INRIA reasercher in the PARIS project of INRIA-Rennes (IRISA). His research activities are centered on architecture, operating systems and compilation techniques for parallel and distributed computing. His current work is focused on the development of a DSM for the implementation of runtime systems on large clusters and for the management of data repositories on the Grid. In the recent past, he participated to the IST POP European project on the implementation of an OpenMP system for clusters using distributed shared memories (DSM). He is currently involved in the XtreemOS European project. The objective of XtreemOS is the development of a Grid operating system with native support for virtual organizations. He is the leader of the Grid'5000 team at INRIA-Rennes.

Stephane Lanteri is a researcher at INRIA Sophia Antipolis in a scientific computing team. His current activities are concerned with the design of unstructured mesh based numerical methods for the discretization of PDE systems modeling wave propagation phenomena, domain decomposition and multilevel algorithms and high performance parallel and distributed computing. He is the scientic coordinator of the Grid5000@Sophia project which defines the contributions of INRIA Sophia Antipolis to the Grid'5000 project.

Julien Leduc is the contractor CNRS Research Engineer, a member of Grid'5000 technical committee and participated in the design of the Grid'5000 grid services architecture. He is the technical manager of the reconfiguration feature of Grid'5000: Kadeploy designer and main developer. Previously, he worked on the Clic clustering distribution, and system administration of several clusters in Grenoble.

Nordine Melab received his Master's, Ph.D. and HDR degrees in computer science, from the Laboratoire d'Informatique Fondamentale de Lille (LIFL, Universite de Lille1). He is an Associate Professor at Polytech'Lille and a member of the OPAC team at LIFL. He is involved in the DOLPHIN project of INRIA Futurs. He is particularly a member of the Steering Committee of the Grid'5000 French Nation-Wide project. His major research interests include parallel and grid computing, combinatorial optimization algorithms and applications and software frameworks.

Dr. Pascale Vicat-Blanc Primet, graduated in Computer Science, is senior researcher (Directrice de Recherche) at INRIA. Her research interests include Distributed and Real-Time Systems, High Performance Grid and Cluster Networking, Active Networks, Internet protocols (TCP/IP), Network Quality of Service. She is leading the RESO team, labelled RESO project of the Institut National de la Recherche en Informatique et Automatique (INRIA) at LIP laboratory in LYON (France). Since 2000, she has been very active in the international and national Grid community. Co-chair of the DataTransport Research Group in the Global Grid Forum, she has co-edited several Grid Networking and Transport protocol GGF documents. She is general co-chair of the International GRIDNETS conference and PFLDNET workshop, member of international conferences steering or program committees and reviewer for international conference and journal in Grids and Networking. She has published her work in more than 60 papers in Grid and Networking journals or conferences. She is member of the steering committee of the French ACI MD DataGRID Explorer (GdX) project, of the ACI Grid Grid5000 project, ANR IGTMD and EU Strep EC-GIN project.

Raymond Namyst received his Ph.D. in computer science from Lille in 1997. He held the position of assistant professor in the Computer Science Departement of the Ecole Normale Superieure of Lyon (1997–2002). In 2002, he joined the Computer Science Laboratory of Bordeaux (LaBRI) where he holds a Professor position. He is the head of the Runtime INRIA research project, devoted to the design of high performance runtime systems for parallel architectures. His main research interests are in parallel computing, thread scheduling on multiprocessor architectures, communications over high speed networks and communications within Grids. He has played a major role in the development of the PM2 software suite. He has written numerous papers about the design of efficient runtime systems. He also serves as the chair of the Computer Science Teaching Department of the University of Bordeaux.

Pascale Vicat-Blanc Primet received a Ph.D. degree in computer science from INSA Lyon, France in 1988. Based at École Normale Supérieure de Lyon (ENS-Lyon), she is currently "directrice de recherche" at INRIA and leads the RESO team-project. This team is specialized in communication protocols and software optimization for high-speed networks. Pascale's research interests include high-performance Grid and cluster networking, active networks, TCP, QoS, bandwidth sharing and security. She was a co-chair of the GGF Data Transport Research Group. She is member of the Steering Committee the Gridnets and Pfldnet conferences. Member of the GRID'5000 project steering committee, she is co-charing its Lyon's site. She has published about hundred papers in distributed computing and networking journals and conferences.

Benjamin Quetier is a Ph.D. student of Franck Cappello (INRIA) and works in the Grand-Large project and in the European project CoreGrid. He is also involved in the Grid'5000 project working on virtualization. The goal of his thesis is to build a large scale emulator platform (more than 100 K nodes) over Grid'5000. The first part of his thesis was a comparison of the diverse virtualization tools such as Xen or VMware. He works on the comparison of applications on a real live platform and on a emulated one.

Olivier Richard is an associate professor at the IDIMAG laboratory. He graduated from Paris XI University with a Ph.D. in computer science in 1999. His researh interests are focused on system architecure for high performance computing and large distributed system (cluster, Grid and P2P). His is co-leader of OAR and Kadeploy software projects.

El-Ghazali Talbi received his Master's and Ph.D. degrees in computer science, both from the Institut National Polytechnique de Grenoble. He is presently Professor in computer science at Polytech'Lille (Universite de Lille1), and researcher in Laboratoire d'Informatique Fondamentale de Lille. He is the leader of OPAC team at LIFL, the DOLPHIN project at INRIA Futurs and the platform of bioinformatics of Lille (Genopole de Lille). He took part to several CEC Esprit and national research projects. His current research interests are mainly parallel and grid computing, combinatorial optimization algorithms and applications and software frameworks.

Iréa Touche took part in the project e-toile, which was the first major French high performance data transfer grid project, when she studied to obtain her engineering degree in applied mathematics and scientific calculations. She was in charge of a part of the cluster's configuration of the CEA (Commissariat à l'energie atomique) and also had to deploy a parallel application of molecular dynamics called CHARMM. After that, she worked for 6 months at IRIT (Institut de Recherche en Informatique de Toulouse), on the Grid'5000 project. She had to "gridify" four applications used by different laboratories of INP Toulouse (Institut National Polytechnique). To do this she studied the application's features, and deployed them on one or more clusters of the grid. For the multi-parametric applications, she used several sites in order to be able to easily carry out a great number of executions. For the parallel ones, she has made scalability tests. She is currently working at the LGC (Laboratoire de Génie Chimique), where she helps researchers to optimize their scientific codes.

Note

1 http://ina2.eivd.ch/Collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html

References

Augerat, P., Martin, C., and Stein, B. 2002. Scalable monitoring and configuration tools for grids and clusters. *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*. IEEE Computer Society.

Bolze, R., Caron, E., and Desprez, F. 2006. A monitoring and visualization tool and its application for a network enabled server platform. In LNCS, editor, *Parallel and Distributed Computing Workshop of ICCSA 2006*, 8–11 May, Glasgow, UK.

Caron, E. and Desprez, F. 2006. DIET: A scalable toolbox to build network enabled servers on the Grid. *International Journal of High Performance Computing Applications*, 20(2):335–352.

Casanova, H., Legrand, A., and Marchal, L. 2003. Scheduling distributed applications: the simgrid simulation framework. *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, Tokyo, Japan.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. 2003. PlanetLab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12.

Daydé, M., Giraud, L., Hernandez, M., L'Excellent, J.-Y., Pantel, M., and Puglisi, C. 2004. An overview of the grid-tlse project. *Proceedings of 6th International Meeting VECPAR 04*, June, Valencia, Spain.

Dumitrescu, C. and Foster, I. 2005. Gangsim: A simulator for grid scheduling studies. *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)*, May, Cardiff, UK.

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

Georgiou, Y., Richard, O., Neyron, P., Huard, G., and Martin, C. 2005. A batch scheduler with high level components. *Proceedings of CCGRID'2005*, May, Cardiff, UK. IEEE Computer Society.

Liu, X., Xia, H., and Chien, A. 2004. Validating and scaling the MicroGrid: A scientific instrument for grid dynamics. *The Journal of Grid Computing* 2(2):141–161.

Melab, N. 2005. *Contributions a la resolution de problemes d'optimisation combinatoire sur grilles de calcul*. Ph.D. thesis, November, LIFL, USTL.

Mezmaz, M., Melab, N., and Talbi, E.-G. 2006. A grid hybrid exact approach for solving multi-objective problems. *Proceedings of the 9th IEEE/ACM International Workshop on Nature Inspired Distributed Computing (NIDISC'06 – in conjunction with IPDPS'2006)*, Rhodes Island, Greece.

Ponish, A., Azzaro-Pantel, C., Domenech, S., and Pibouleau, L. 2005. About the relevance of mathematical programming and stochastic optimisation methods: application to the optimal batch plant design problems. *ESCAPE 15*, May 29–June 1.

Ruiz, R. and Stutzle, T. 2004. A simple and effective iterative greedy algorithm for the flowshop scheduling problem. Technical Report, European Journal of Operational Research, in print.

Seymour, K., Lee, C., Desprez, F., Nakada, H., and Tanaka, Y. 2004. The end-user and middleware APIs for GridRPC. *Workshop on Grid Application Programming Interfaces*, in conjunction with GGF12, September, Brussels, Belgium.

Takefusa, A., Matsuoka, S., Aida, K., Nakada, H., and Nagashima, U. 1999. Overview of a performance evaluation system for global computing scheduling algorithms. *HPDC '99: Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing*, Washington, DC, USA. IEEE Computer Society.

White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. 2002. An integrated experimental environment for distributed systems and networks. *OSDI02 Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, pp. 255–270, December, Boston, MA.