Hybrid Distributed Computing Infrastructure Experiments in Grid5000: Supporting QoS in Desktop Grids with Cloud Resources

Simon Delamare, Gilles Fedak and Oleg Lodygensky

simon.delamare@inria.fr - gilles.fedak@inria.fr - lodygens@lal.in2p3.fr

Abstract

Hybrid Distributed Computing Infrastructures (DCIs) allow users to combine Grids, Desktop Grids, Clouds, etc. to obtain for their users even larger computing capabilities. In this paper, we present an experimental study of the SpeQuloS framework which aims at providing QoS to Desktop Grid by provisioning on-demand Cloud resources. We describe the experimental platform which relies on Grid5000 to mimic both a Desktop Grid system and a Cloud system. Preliminary results are presented which shows the potential of the SpeQuloS approach.

1 Introduction

There is a growing demand for computing power from scientific communities to run their applications and process large volumes of scientific data. Meanwhile, the supply of Distributed Computing Infrastructures (DCIs) for scientific computing continues to diversify. Users can not only select their preferred architectures amongst Supercomputers, Clusters, Grids, Clouds, Desktop Grids and more, based on parameters such as performance, reliability, cost or quality of service, but can also combine transparently several of these infrastructures.

As an example of such hybrid DCIs, the European FP7 projects EDGeS[1] and EDGI[2] have developed bridge technologies to make Desktop Grid (DG) systems, such as BOINC[3] or XtremWeb-HEP[4] (XWHEP) transparently available to any Enabling Grids for E-sciencE[5] (EGEE) grid users as a regular cluster. Indeed, other attempts have successfully built systems where Grid infrastructures are supplemented by low cost volunteer computers[6] or by Cloud resources[7], benefiting from the elastic resource provisioning, to meet users' peak demands.

In this paper, we present experiments conducted on Grid5000 to address a particular scenario where the objective is to provide Quality of Service (QoS) capabilities for the DG part of the EDGeS DCI. We propose the SpeQuloS framework to supplement DGs with Cloud resources in order to enhance the QoS of applications executed on DG. Providing QoS features in Grids is hard and not solved yet satisfactorily and is even more difficult in an environment where there are no guaranteed resources. In DG systems, resources can leave the system at any time for a long time or forever even after taking several work-units with the promise of computing them.

We describe the SpeQuloS architecture and its main components. SpeQuloS features multi-DG support (BOINC, XWHEP), multi Cloud services support (EC2, Eucalyptus). Next, we describe our testbed, called XtremG5K, which links production Grid (EGEE) and experimental Grid (Grid5000) through an XWHEP server and we report on preliminary experimental results. Finally, we describe the Grid5000 libcloud[8] driver that we developed to start VM instances on Grid5000 as with any commercial Clouds.

2 SpeQuloS

SpeQuloS is a framework to provide Quality of Service (QoS) to users of unreliable distributed computing infrastructures (DCI), such as Desktop Grids (DG) based on volunteer computing, by provisioning stable resources from Cloud services. In this paper, the QoS is described by Bag of Tasks (BoT) response time, which is the time from BoT submission to results delivery. We denote as a BoT a collection of work units, or individual jobs submitted by user to a DCI. The main objective of SpeQuloS is to bring to users some information on their BoT response time and to decrease this time by provisioning resources from Cloud services if they so requested.

To decrease a BoT response time, SpeQuloS adds dedicated resources to its computation. SpeQuloS uses Infrastructure as a Service (IaaS) Clouds to provide those resources. To supply resources to a DG, SpeQuloS instantiates a virtual machine, called Cloud worker, starts some DG worker software, makes it join the DG server where the BoT is computed and stops it when necessary. As access to Cloud resources is limited, SpeQuloS provides a framework to share those resources between users and account their utilization. To use Cloud resources in a smart way, SpeQuloS gathers information about the QoS in a DG. This information is used to evaluate the current level of QoS and to anticipate benefits of allocating cloud resources.

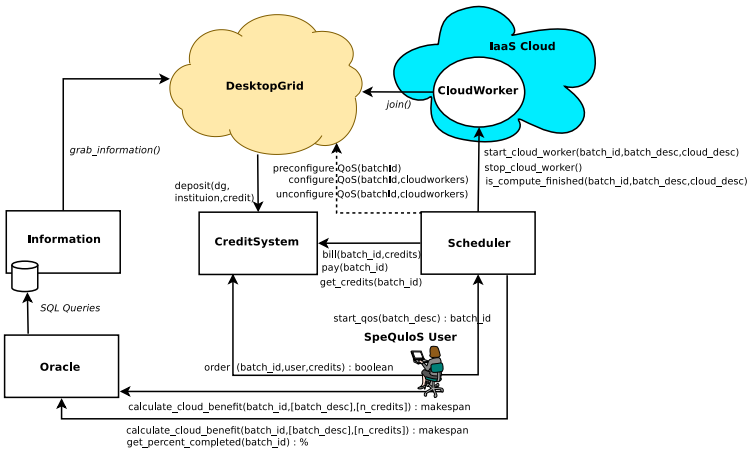


Figure 1: The SpeQuloS modules and their interactions

SpeQuloS is composed of several modules as shown in figure 1:

- The Credit System module is in charge of accounting the Cloud-related operations. It interacts with users to receive orders to provision a BoT with Cloud resources and with the Scheduler module to bill their utilization.
- The Information module gathers and stores information relevant to QoS from DG, such as the BoT completion (which denotes the ratio between the number of completed and uncompleted jobs) and the number of workers participating in the DG.
- The Oracle module makes some prediction about QoS in the DG. Using the information provided by the Information module, the Oracle computes an estimated response time for a BoT. The response time is estimated with and without use of Cloud resources, to point out its benefit. The main role of the Oracle module is to give information to users who take decision to spend Cloud resource for their BoT.
- The Scheduler module manages the BoT and the Cloud workers during the system execution. When QoS are requested by an user, it instantiates some Cloud workers and assigns them to the appropriate BoT. The module declares to the Credit System the Cloud resources spent. It also interacts with the DG server, to perform some configuration related to BoT and Cloud workers, which depends on the type of DG considered.

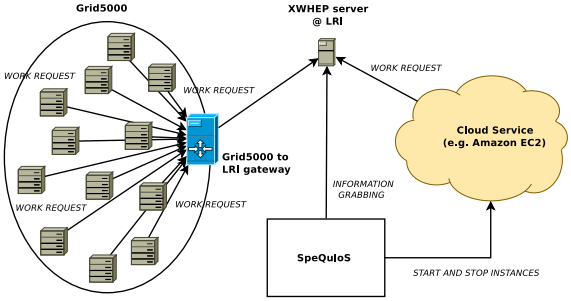
Each module is independent and can be deployed on different computers. Python programming language and MySQL databases are used for their implementation. Communications between modules use web services. SpeQuloS does not depend on a particular DG technology. Currently, SpeQuloS supports BOINC and XWHEP DG. It does not depend on a particular Cloud service either. It uses the libcloud[8] Application Programming Interface (API) as an abstraction layer to access to various types of Cloud services transparently. Cloud services tested include Amazon EC2 and Eucalyptus, and in the next future we will consider OpenNebula and Nimbus.

3 Using Grid5000 as Desktop Grid workers

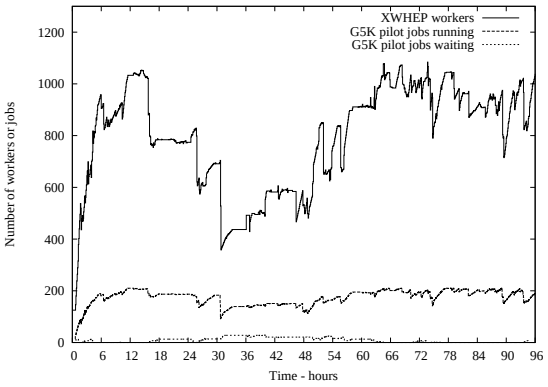
To test SpeQuloS and evaluate its performance under realistic conditions, we performed experimentations using Grid5000. In DGs, from few hundred to several thousand worker nodes are participating to computational effort during several days. To reproduce this situation, we used Grid5000 resources as DG workers. In this section, we are presenting this experimentation and its results.

3.1 SpeQuloS and the XtremG5K project

Our experimentation is supported by the XtremG5K project, which main goal is to use Grid5000 to provide computational resources to an XWHEP DG. The XtremG5K project is composed of several components:



(a) The XtremG5K project and SpeQuloS architecture



(b) Number of XWHEP workers and Grid5000 pilot jobs running and waiting during the experimentation

Figure 2: The XtremG5K project

- An XWHEP server, hosted at Laboratoire de Recherche en Informatique (LRI). It can accept jobs submission from EGEE users through the 3Gbridge[9] middleware. Unfortunatelly, at the time of the experiment, the 3Gbridge was not functionning and we had to use the XWHEP server as an entry point. However, this does not impact the SpeQuloS scenarios presented here. The XWHEP server is in charge of jobs distribution among XWHEP workers, and results collection.
- A gateway, administrated in Grid5000, which enables communications between Grid5000 network and the LRI XWHEP server thanks to XWHEP ability to communicate through a proxy since version 7.3.0.
- A set of XWHEP workers, executed on Grid5000 nodes. In the experimentation presented in this paper, a Grid5000 job is considered as a pilot job which runs one or several XWHEP workers. The XWHEP worker software relies on Java plateform to run and does not need to use a specific Grid5000 environment. To approximate a voluntary-based and best effort participation to computation as observed with DG worker nodes, we decided to use the "besteffort" queue of Grid5000. We can thus reproduce DG caracteristics we are interested in: Variable amount of computing resources with unpredictable and unanounced nodes departure.

In this experimentation, SpeQuloS was connected to the LRI XWHEP server, and to the Amazon EC2 Cloud service. It does not need to communicate with Grid5000. The diagram 2a shows the experimentation architecture.

3.2 Grid5000 resources usage

The experimentation has been deployed on seven Grid5000 sites (Lyon, Grenoble, Bordeaux, Lille, Nancy, Sophia and Toulouse). On each site, we ran the algorithm 1 for pilot job submission. Each pilot job, denoted as "start_XWHEP_workers" in the algorithm, is submitted to one Grid5000 node and starts one XWHEP worker per CPU core on the node executing it.

We ran this experimentation during 4 days. The figure 2b shows the number of pilot jobs, running or waiting for submission in Grid5000, and the corresponding number of XWHEP workers, according to time of experimentation. We can see that the total number of Grid5000 pilot jobs was comprised between 100 and 200, which lead 400 to more than 1000 workers to participate to the DG. The number of Grid5000 pilot jobs waiting for submission on all sites simultaneously never exceeds 30.

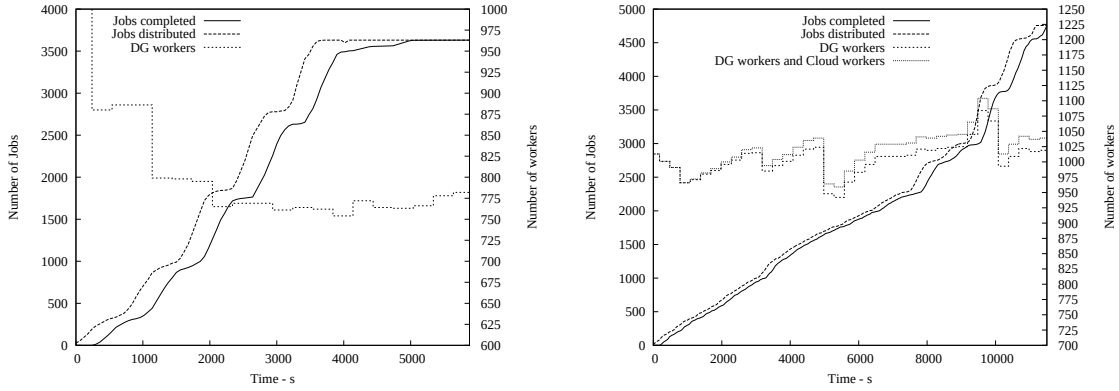
3.3 Experimentation results

We ran several experimentation scenarios with or without using SpeQuloS. In each scenario, an XWHEP user submits a BoT containing from 1000 to 10000 jobs. Each individual job needs few minutes to be computed by a worker. Jobs are distributed by the LRI XWHEP server to worker nodes hosted in Grid5000 and to Cloud workers from Amazon EC2 when SpeQuloS was used. In this section, we are presenting the results of two of these scenarios.

Figure 3a shows result from a scenario where SpeQuloS is not used. A BoT of 3630 jobs is submitted at the beginning of the scenario. The figure shows the number of jobs distributed and completed as well as the number of DG workers, according to the scenario time. The number of jobs distributed denotes the

Algorithm 1 Algorithm for pilot jobs submission executed on each Grid5000 site

```
max_#p jobs ← 30
max_#p jobs_waiting ← 7
while true do
  if current_#p jobs < max_#p jobs then
    if current_#p jobs_waiting < max_#p jobs_waiting then
      "Submit start_XWHEP_workers to one Grid5000 node in besteffort queue"
    else
      "Too many pilot jobs waiting"
    end if
  else
    "Maximum number of pilot jobs reached"
  end if
  sleep(15 minutes)
end while
```



(a) SpeQuloS not used

(b) SpeQuloS used

Figure 3: Submission and completion of jobs with number of desktop grid workers, according to the scenario time

number of jobs that have been sent to a DG worker. The number of jobs completed denotes the number of jobs for which execution by DG node has been completed, and results sent back to the LRI server.

We can see that the BoT completion increases quickly during the first 4000th seconds of the scenario. Then, the completion grows slowly while all jobs have been distributed to workers. This figure illustrates the "tail effect" of BoT completion in DG, caused by jobs submitted to DG workers who then leave the DG. In that case, the DG server has to resubmit the job to another worker, but the time required for this operation can be significant, as the missing workers have to be detected first. In our experiments, the tail effect is caused by preempted Grid5000 Best Effort jobs, causing XWHEP workers to leave without noticing the server.

By provisioning stable resources from Cloud, one goal of SpeQuloS is to address this problem. Figure 3b shows similar results from a scenario where SpeQuloS was used. In this scenario, a BoT of 4750 jobs was submitted. In addition to previous scenario results, the figure shows the Cloud workers started by SpeQuloS to participate to the DG. The amount of cloud resources available to SpeQuloS was limited and equivalent to 60 CPU.hours of the "small" instance of Amazon EC2.

No tail effect can be observed in this scenario. The BoT completion increases regularly during the first 8000th seconds of the scenario and grows even faster afterwards. We can see that SpeQuloS, according to the policy used in its Scheduler module, starts some Cloud workers to participate to the DG. The first worker is started at the 15th minute of the scenario. SpeQuloS then adds Cloud workers gradually until the maximum of 20 simultaneous workers is reached.

These experimentations are not sufficient to draw any general conclusion about the ability of SpeQuloS to mitigate the tail effect. A lot of parameters still have to be investigated, and some reproducible set of experiments have to be conducted. However, the experimentations presented in this section demonstrate the abil-

Table 1: State of the Grid5000 libcloud driver implementation. Implemented features are marked Ok and unimplemented features are marked WIP (Work In Progress)

Provider	Create Instance	Reboot Instance	Destroy Instance	List Disk Images	Deploy Disk Images	List Node Size
Grid5000	Ok	Ok	Ok	WIP	WIP	WIP

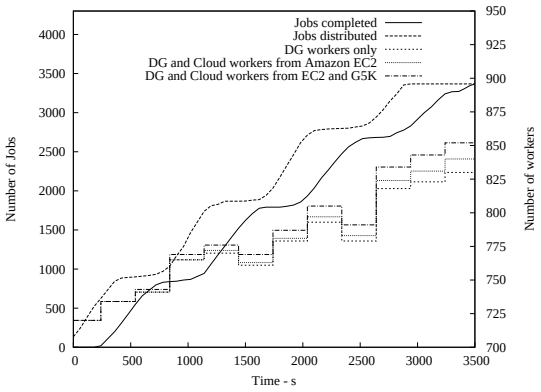


Figure 4: Submission and completion of jobs with number of desktop grid workers, according to the scenario time. Grid5000 resources are used as Cloud workers

ity to use Grid5000 to run large scale experiments of hybrid infrastructure, mixing DG and Cloud services.

4 Using Grid5000 as a Cloud

To evaluate the benefits of Cloud resources provisioning in a large scale scenario, a large amount of resources from Cloud must be available to experiments. However, those resources are not always available to researchers because of cost of public Cloud services such as Amazon EC2 and complexity to build and maintain private Cloud services using technologies such as Eucalyptus or OpenNebula. To address this issue, we decided to use Grid5000 as a new type of Cloud available to SpeQuloS. Indeed, Grid5000 can provide on-demand computing resources and includes most of features of common Cloud services.

4.1 The Grid5000 libcloud driver

As SpeQuloS uses the libcloud API as a common interface to interact with various types of Cloud services, we decided to use the Grid5000 API[10] to implement a new libcloud <<driver>> to access to Grid5000 resources. This implementation allows to interact with Grid5000 as a typical IaaS Cloud service, with the same API as any other Cloud service for which driver exists in libcloud. Our implementation uses Grid5000 node reservation and environment deployment to reproduce IaaS Cloud features and does not involve any host virtualization technology. The current state of this implementation is shown is table 1. It is currently possible to interact with a Grid5000 Cloud instances (Create Instance, Reboot Instance and Destroy Instance features) but only using the default Grid5000 disk image (List Disk Image and Deploy Disk Image features) without dealing with nodes specifications, such as number of CPU (List Node Size feature). The implementation of those missing features is underway.

With the Grid5000 libcloud driver, a large amount of Cloud resources is made available to researchers. Software which use the libcloud API can benefit of Grid5000 resources without any additional development. It is also possible to combine Grid5000 and other Cloud services in the same experiment.

4.2 Experimentation results

To validate the Grid5000 libcloud driver, we ran an experimentation scenario with SpeQuloS provisioning Cloud resources from both Amazon EC2 and Grid5000. The rest of the experiment environment is similar to that presented in section 3.

Figure 4 shows results from a scenario where SpeQuloS uses, in addition to Amazon EC2 resources, some Grid5000 resources as Cloud workers. In this scenario, up to 15 Grid5000 Cloud workers are started

by SpeQuloS to participate to the DG, each Cloud worker running on a Grid5000 node hosted in the Rennes site.

This experiment exhibits the ability to use Grid5000 as an IaaS Cloud service, thanks to the Grid5000 libcloud driver implementation.

5 Conclusion and Future Work

In this paper, we presented some experiments using Grid5000 to evaluate an hybrid DCI, which main goal is to bring QoS features to DG users by provisioning resources from Cloud services. We presented SpeQuloS, the framework we developed for this purpose. We described the XtremG5K project, which enables Grid5000 resources to be connected to an EGEE XWHEP server, and shown some preliminary results for this project and SpeQuloS. We also presented the Grid5000 libcloud driver, an extension of the libcloud API to interact with Grid5000 like a classic Cloud service.

We will complete the Grid5000 libcloud driver implementation as future work. To improve the SpeQuloS evaluation, we also plan to conduct some reproducible scenarios in Grid5000, based on realistic worker availability traces. We will also deploy SpeQuloS in scenarios involving other type of DG, such as BOINC.

6 Acknowledgment

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

We thank l'Institut des Grilles for having supported our project XtremG5K through the interface projects between production grid and experimental grid.

SpeQuloS and EDGI (European Desktop Grid Initiative) receives Community funding from the European Commission within Research Infrastructures initiative of FP7 (grant agreement Number RI-261556).

References

[1] Etienne Urbah, Peter Kacsuk, Zoltan Farkas, Gilles Fedak, Gabor Kecskemeti, Oleg Lodygensky, Attila Marosi, Zoltan Balaton, Gabriel Caillat, Gabor Gombas, Adam Kornafeld, Jozsef Kovacs, Haiwu He, and Robert Lovas. Edges: Bridging egee to boinc and xtremweb. *Journal of Grid Computing*, 7(3):335–354, September 2009.

[2] Edgi: European desktop grid initiative project. `http://edgi-project.eu`, 2011.

[3] David Anderson. BOINC: A System for Public-Resource Computing and Storage. In *proceedings of the 5th IEEE/ACM International GRID Workshop*, Pittsburgh, USA, 2004.

[4] Franck Cappello, Samir Djilali, Gilles Fedak, Thomas Herault, Frédéric Magniette, Vincent Néri, and Oleg Lodygensky. Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid. *Future Generation Computer Systems*, 21(3):417–437, mar 2005.

[5] Fabrizio Gagliardi, Bob Jones, François Grey, Marc-Elian Bégin, and Matti Heikkurinen. Building an infrastructure for scientific grid computing: Status and goals of the egee project. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 363(1833):1729–1742, 2005.

[6] Mark Silberstein, Artyom Sharov, Dan Geiger, and Assaf Schuster. Gridbot: Execution of bags of tasks in multiple grids. In *SC '09: Proceedings of the 2009 ACM/IEEE conference on Supercomputing*, New York, NY, USA, 2009. ACM.

[7] P. Marshall, K. Keahey, and T. Freeman. Elastic site: Using clouds to elastically extend site resources. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010)*, Melbourne, Australia., May 2010.

[8] libcloud, a unified interface to the cloud. `http://incubator.apache.org/libcloud/`, 2011.

[9] Haiwu He, Gilles Fedak, Peter Kacsuk, Zoltan Farkas, Zoltan Balaton, Oleg Lodygensky, Etienne Urbah, Gabriel Caillat, and Filipe Araujo. Extending the EGEE Grid with XtremWeb-HEP Desktop Grids. In *Proceedings of CCGRID'10, 4th Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid 2010)*, pages 685–690, Melbourne, Australia, May 2010.

[10] Grid'5000 API. `https://api.grid5000.fr/`, 2011.