HAL
open science

Advanced QoS Prototype for the EDGI Infrastructure

Simon Delamare, Gilles Fedak, Derrick Kondo, Oleg Lodygensky, Péter Kacsuk, Jozsef Kovacs, Filipe Araujo

To cite this version:

Simon Delamare, Gilles Fedak, Derrick Kondo, Oleg Lodygensky, Péter Kacsuk, et al.. Advanced QoS Prototype for the EDGI Infrastructure. [Research Report] RR-8295, INRIA. 2013. hal-00819907

HAL Id: hal-00819907

https://inria.hal.science/hal-00819907v1

Submitted on 2 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Advanced QoS Prototype for the EDGI Infrastructure

Simon Delamare, Gilles Fedak, Derrick Kondo, Oleg Lodygensky, Peter Kacsuk, Jozsef Kovacs, Filipe Araujo

RESEARCH REPORT
N° 8295
April 2013
Project-Team GRAAL

ISSN 0249-6399 ISRN INRIA/RR--8295--FR+ENG

Advanced QoS Prototype for the EDGI Infrastructure

Simon Delamare*, Gilles Fedak†, Derrick Kondo‡, Oleg Lodygensky§, Peter Kacsuk¶, Jozsef Kovacs‖, Filipe Araujo**

Project-Team GRAAL

Research Report n° 8295 — April 2013 — 41 pages

Abstract:
This document provides the second deliverable of EDGI JRA2. It is produced by the INRIA team, the SZTAKI team, the LAL/IN2P3 team and the University of Coimbra team. This document aims at describing achievements and results of JRA2 tasks "Advanced QoS Scheduler and Oracle" and "Support In Science Gateway".
Hybrid Distributed Computing Infrastructures (DCIs) allow users to combine Grids, Desktop Grids, Clouds, etc. to obtain for their users large computing capabilities. The EDGI infrastructure belongs to this kind of DCIs. The document presents the SpeQuloS framework to provide quality of service (QoS) for application executed on the EDGI infrastructure. It also introduces EDGI QoS portal, an user-friendly and integrated access to QoS features for users of EDGI infrastructure.
In this document, we first introduce new results from JRA2.1 task, which collected and analyzed batch execution on Desktop Grid. Then, we present the advanced Cloud Scheduling and Oracle strategies designed inside the SpeQuloS framework (task JRA2.2). We demonstrate efficiency of these strategies using performance evaluation carried out with simulations. Next, we introduce Credit System architecture and QoS user portal as part of the JRA2 Support In Science Gateway (task JRA2.3). Finally, we conclude and provide references to JRA2 production.

Key-words: Distributed Computing, Quality of Service, Bag of Tasks

* INRIA/University of Lyon - Simon.Delamare@inria.fr
† INRIA/University of Lyon - Gilles.Fedak@inria.fr
‡ INRIA/University of Joseph Fourier - Derrick.Kondo@inria.fr
§ IN2P3/University of Paris XI - Oleg.Lodygensky@lal.in2p3.fr
¶ MTA/STZAKI - peter.kacsuk@sztaki.mta.hu
‖ MTA/STZAKI - kovacs.jozsef@sztaki.mta.hu
** University of Coimbra - filipius@dei.uc.pt

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Prototype avancé assurant la Qualité de Service à l'infrastructure EDGI

Résumé : Ce document fournit le deuxième livrable pour la tâche JRA2 du projet européen European Desktop Grid Initiative (FP7 EDGI). Il est produit par les équipes de l'INRIA, de SZTAKI, du LAL/IN2P3 et de l'Université de Coimbra. Ce document vise à décrire les réalisations et les résultats qui concernent la qualité de service pour l'infrastructure de grilles de PCs européenne EDGI.

Mots-clés : Calcul distribué, Qualité de Service, Sac de Tâches

Project number: RI 261556

Project acronym: EDGI

Project full title: European Desktop Grid Initiative

Research Infrastructures
INFRA-2010-1.2.1 Distributed computing infrastructures
DCI-1.2.1.3 Middleware and repositories

D7.2 - Advanced QoS prototype for EDGI infrastructure

Due date of deliverable: 31/01/2012 Actual submission date: 31/01/2012

Start date of project: 01/06/2010 Duration: 24 months

INRIA

Dissemination level: PU

Version: 1.0

EDGI is supported by the FP7 Capacities Programme under contract nr RI 261556

Contents

1 Glossary **6**

2 About this document **7**

3 Summary from D7.1 **8**

4 New results from task JRA2.1 (QoS Information) **10**

5 Task JRA2.2: Advanced QoS Scheduler and Oracle **13**
5.1 QoS Oracle . . . 13
5.1.1 Cloud Deployment Strategies . . . 13
5.2 QoS Scheduler . . . 14
5.2.1 Cloud Workers Management . . . 14
5.3 Performance Evaluation . . . 14
5.3.1 Evaluation Setup . . . 16
5.3.2 QoS Strategies Evaluation . . . 16
5.3.3 Cloud Resource Consumption . . . 18
5.3.4 Completion Time Speed Up . . . 18
5.3.5 Prediction Accuracy . . . 20
5.4 Conclusion on Performances . . . 20

6 Task JRA2.3: QoS Support In Science Gateway **22**
6.1 Credit System . . . 22
6.1.1 Credit System Cloud Resources Billing . . . 23
6.1.2 Credit System Earning Policy . . . 23
6.2 QoS Portal . . . 25
6.2.1 About WS-PGRADE/gUSE portal . . . 25
6.2.2 Integrated functionalities . . . 25
6.2.3 Ordering credits for QoS support . . . 25
6.2.4 Get available credits . . . 27
6.2.5 Get current QoS orders . . . 28
6.2.6 Get information on a batch completion . . . 29
6.2.7 Make prediction about batch completion time . . . 30
6.2.8 Availability . . . 31

7 New Software Development **32**
7.1 Cloud Workers Management . . . 32
7.2 Supporting XWHEP . . . 33
7.2.1 Support for Information Module . . . 33
7.2.2 Support for Credit System Module . . . 33
7.2.3 Cloud Duplication Strategy . . . 33
7.3 Supporting BOINC . . . 35
7.3.1 Support for Information Module . . . 35
7.3.2 Support for Credit System Module . . . 35
7.3.3 Reschedule Strategy . . . 35
7.4 Deployment in Development Infrastructure . . . 36

8 Conclusion **37**
8.1 Publications 37
8.2 Measures/indicators 38

A JRA2 D7.2 ressources **39**
A.1 SpeQuloS package 39
A.2 SpeQuloS documentation 39
A.2.1 SpeQuloS installation documentation 39
A.2.2 SpeQuloS usage documentation 39
A.2.3 Cloud Worker image creation documentation 39
A.2.4 BOINC server patching documentation 39
A.2.5 Cloud Duplication for XWHEP 39
A.3 QoS Portal resources 39

1 Glossary

BOINC	Berkeley Open Infrastructure for Network Computing
BoT	Bag of Task, or batch
DG	Desktop Grids: Grids composed of Desktop PCs.
EDGeS	Enabling Desktop Grids for E-Science, the FP7 project that aims to bridge Service Grids and Desktop Grids
INRIA	Institut National de Recherche en Informatique et en Automatique (Saclay, France)
LAL	Laboratoire de l'accelerateur Lineaire (Orsay, France) belonging to IN2P3
LRI	Laboratoire de Recherche en Informatique (Orsay, France)
QoS	Quality of Service
SG	Service Grids: Grids composed of resources belonging to institutions.
SpeQuloS	JRA2 software to support QoS in Desktop Grids
SQL	Structured Query Language
WN	Worker Node
WU	Workunit
XW	XtremWeb
XWHEP	XtremWeb for High Energy Physics: Grid middleware developed by IN2P3 LAL (from XtremWeb of INRIA) and used inside the EDGI project

2 About this document

This document provides the second deliverable of EDGI JRA2. It is produced by the INRIA team, the SZTAKI team, the LAL/IN2P3 team and the University of Coimbra team. This document aims at describing achievements and results of JRA2 tasks "Advanced QoS Scheduler and Oracle" and "Support In Science Gateway".

Hybrid Distributed Computing Infrastructures (DCIs) allow users to combine Grids, Desktop Grids, Clouds, etc. to obtain for their users large computing capabilities. The EDGI infrastructure belongs to this kind of DCIs. The document presents the SpeQuloS framework to provide quality of service (QoS) for application executed on the EDGI infrastructure. It also introduces EDGI QoS portal, an user-friendly and integrated access to QoS features for users of EDGI infrastructure.

In this document, we first introduce new results from JRA2.1 task, which collected and analyzed batch execution on Desktop Grid. Then, we present the advanced Cloud Scheduling and Oracle strategies designed inside the SpeQuloS framework (task JRA2.2). We demonstrate efficiency of these strategies using performance evaluation carried out with simulations. Next, we introduce Credit System architecture and QoS user portal as part of the JRA2 Support In Science Gateway (task JRA2.3). Finally, we conclude and provide references to JRA2 production.

3 Summary from D7.1

The European FP7 projects EDGeS[26, 3, 18, 27, 28, 2, 29] and EDGI[4] have developed bridge technologies to make Desktop Grid (DG) systems, such as BOINC[5] or XtremWeb-HEP[30, 31, 16, 17, 32] (XWHEP) transparently available to any Enabling Grids for E-sciencE[7] (EGEE) grid users as a regular cluster. Indeed, other attempts have successfully built systems where Grid infrastructures are supplemented by low cost volunteer computers[8] or by Cloud resources[9], benefiting from the elastic resource provisioning, to meet users' peak demands.

The main problem with the current EDGI infrastructure is that it cannot give any QoS support for those applications that require a faster execution in the public DG part of the infrastructure. For example, a public DG system enables clients to return work-unit results in the range of weeks. There is a clear need for the infrastructure to support applications which require a lower latency.

The approach followed by JRA2 work package to improve QoS management is to develop the SpeQuloS framework for the EDGI infrastructure which is composed of four main components: QoS Information, QoS Credit System, QoS Scheduler and a QoS Oracle. The QoS Scheduler component is responsible to provision additional Cloud resources for a certain application if QoS requirements need it and if the user has already collected the necessary number of credits to support its application. The QoS Oracle implements the strategies to decide when, how and how many Cloud resources should be provisioned. The QoS Credit System mandates the Cloud resources between the users. Finally, the monitoring system is developed in order to store QoS Information and to provide a repository of data on DGs' infrastructure workloads.

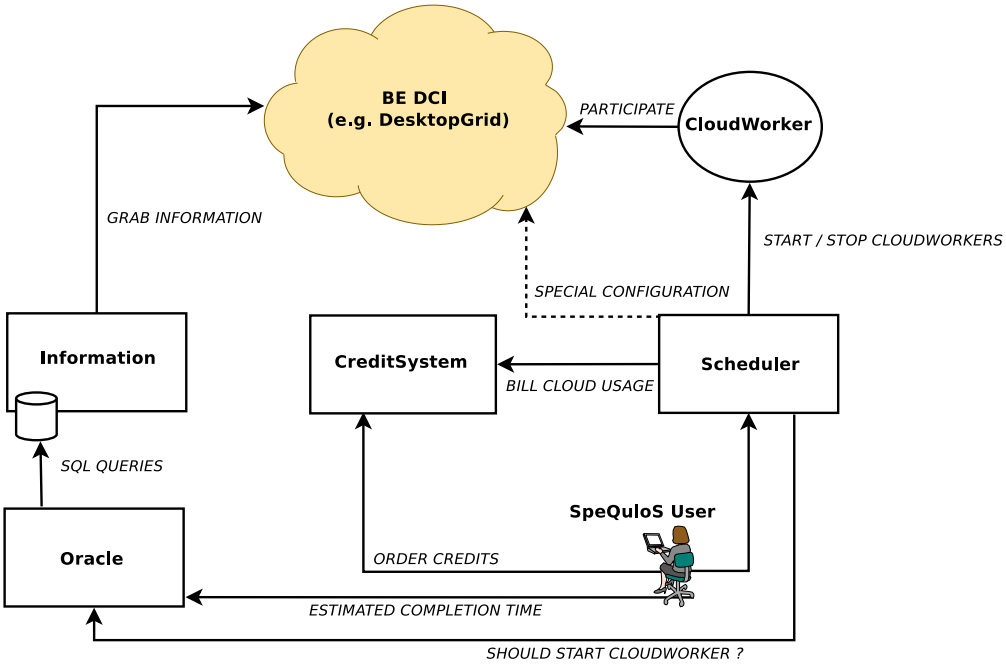


Figure 1: The SpeQuloS modules and their interactions

At stage of D7.1, JRA2 has developed prototypes of the Information, Credits, Oracle and Scheduler and initial support for XWHEP and BOINC DGs and Cloud services, as denoted in figure 1. In the first period of the project, most of effort has focused on designing and providing

the framework. The other main contribution of D7.1 was to collect batch executions on DGs, using monitoring through the Information module as well as other source of data collection. In the second period of the project, we analyzed these traces to characterize the tail effect. In this document, we propose, evaluate and implement advanced QoS strategies to improve QoS for the EDGI infrastructure.

4 New results from task JRA2.1 (QoS Information)

Desktop Grids are an attractive solution to provide computing resources at a low cost. However, the Desktop Grid resource availability can be considered as Best Effort. Usually, computing power is available to the Desktop Grid when the desktop computer is idle. Therefore, a resource becomes unavailable when the computer is used or turned off, sometimes without informing the Desktop Grid server. Volatile nature of Desktop Grid causes a low level of Quality of Service, especially compared to other distributed computing infrastructures (DCIs), such as regular Grids. This is particularly observable for batches executed on DGs, when users wait for all tasks belonging to a batch to be completed.

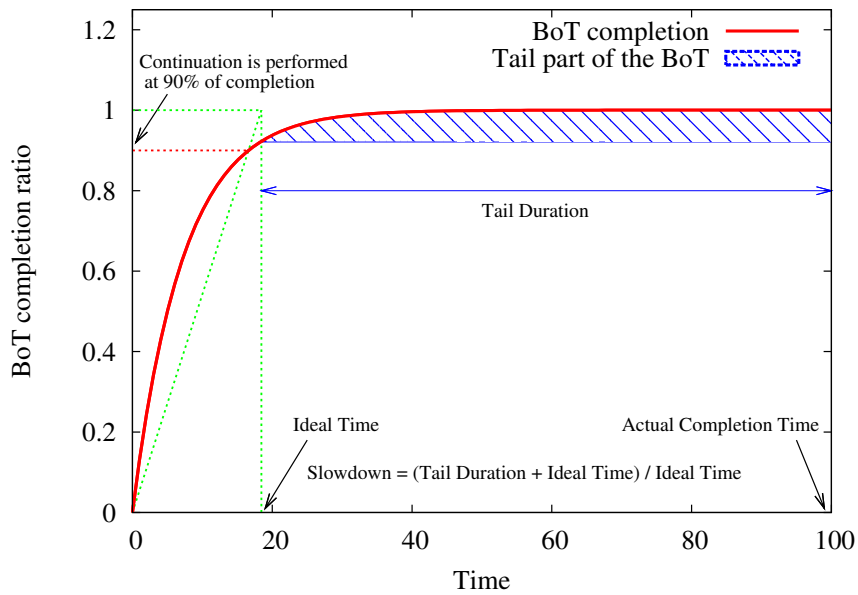


Figure 2: Example of batch execution with Tail Effect

Execution profile of batches observed during QoS Information task is often similar to the one denoted in Figure 2. We have observed that the execution of the last part of a batch takes a large part of the total execution time. We call this event **Tail Effect**. To characterize the Tail effect, we introduce the *tail slowdown*, which is the ratio of an ideal time for batch completion, defined assuming a constant completion rate throughout the execution, versus the actual completion time observed. The ideal time is computed as the obtained by computing the task completion rate at 90% of the BoT execution and by assuming that the completion rate would stay constant up to the end of the execution.

BOINC and XWHEP Desktop Grid middleware have features to handle nodes volatility: resilience to node failures, no need for reconfiguration when new nodes are added, tasks replication, etc. However, observations made from batch execution we captured show that these features are not able to remove the Tail effect. Consequently, QoS delivered to user is low, especially when compared to QoS delivred by SG.

To obtain a better knowledge of the tail effect, we have analyzed some of traces of executions using the result of the QoS Information system deployed in the first year of the project. In addition, we have developped a simulator for BOINC and XWHEP and generated execution

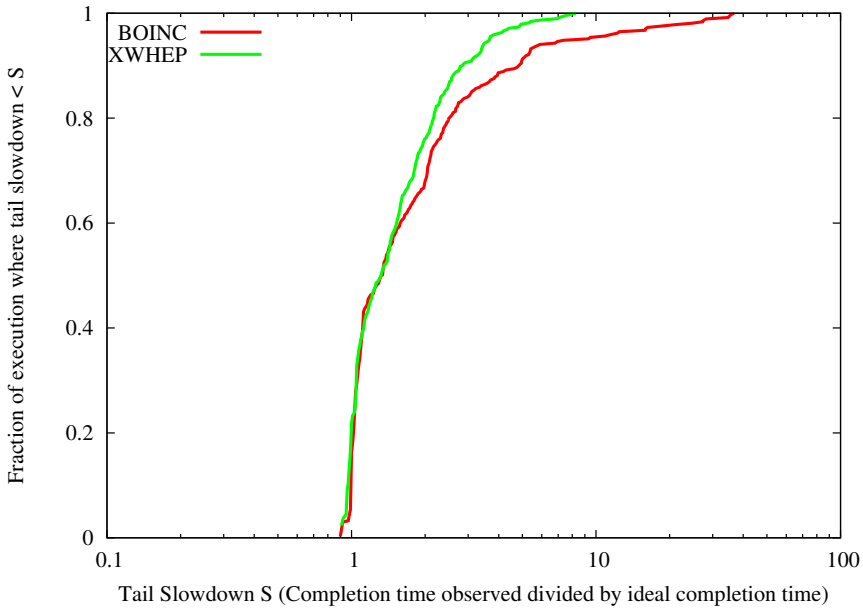


Figure 3: Execution of batches in DGs: Cumulative distribution functions of Tail slowdown. The slowdown denotes the importance of Tail effect by measuring the batch completion time divided by the ideal completion time.

traces on various BoT (see section 5.3 for a detailed presentation of the BoTs and Desktop Grid investigated). Figure 3 presents the cumulative distribution functions of "Tail slowdowns" we measured, for both BOINC or XWHEP middleware of for various BoT workload.

Our observations show that the Tail effect affects about one half of executions, where slowdown is higher than 1.33, meaning that Tail effect slows the execution by more than 33%. Indeed, the Tail effect significantly disturb batch executions in some cases: It doubles the completion time from one quarter (XWHEP) to one third (BOINC) of execution. In worst cases, the slowdown can be as high as 10. More, the large variation in the tail slowdown prevents to predict accurately the Bot execution time. Yet, such prediction is an important objective o improve the usability of the EDGI infrastructure.

Table 1: Average fraction of Bag of Tasks in tail (ratio between the number of tasks in tail and number of tasks in batch) and average percentage of execution time in tail (percentage of total batch execution time spent in the tail)

Avg. % of batch in tail		**Avg. % of time in tail**	
BOINC	**XWHEP**	**BOINC**	**XWHEP**
4.65	5.11	51.8	45.2

Tasks being executed during the Tail effect, i.e. later than the ideal time, create this slowdown by being longer than others to be completed. Table 1 shows the characteristics of the tasks which belongs to the tail, according to middleware used. One can observe that few percents of the tasks are executed in the Tail, but they take an important time of the total execution time.

These results show that the Tail effect causes unsatisfactory level of QoS in Desktop Grids,

for both BOINC and XWHEP middleware. This Tail effect strongly slows batch completion times, and leads to a highly variable and unpredictable execution time.

To mitigate the Tail effect and provide a better QoS to EDGI Desktop Grid users, we propose to supply additional resources to Desktop Grids, to support execution in particular when the Tail effect occurs. Based on our observations, the strategies to decide when and how many Cloud resources should be provisioned must fulfill the following objectives:

- detect when the BoT execution enters the tail
- provision enough Cloud workers to remove the tail slowdown, without over-spending Cloud resources
- decrease the actual BoT completion time as close as possible to the ideal completion time
- improve the quality of the BoT execution time prediction

5 Task JRA2.2: Advanced QoS Scheduler and Oracle

The following section describes the design and performance evaluation of QoS Scheduler and QoS Oracle inside the SpeQuloS framework that has been addressed as JRA2.2 task.

5.1 QoS Oracle

One major feature to enhance users' perceived QoS is to be able to predict batches completion time. By analyzing the current state of a batch execution and the history of previous ones from data collected by the Information module, the Oracle computes the predicted completion time of a batch. If the predicted time is attractive enough for the user, it may decide to order QoS to SpeQuloS.

SpeQuloS Oracle uses the following prediction methods: When called by an user for a prediction, the Oracle gets from the Information module the batch completion ratio (r) and the elapsed time since batch submission ($t_c(r)$). The predicted completion time t_p is computed as follows:

$$t_p = \alpha . \frac{t_c(r)}{r}$$

where α is an adjustment factor which is determined according to the previous batch executions in a given Desktop Grid. It is set to 1 at initialization, and adjusted after each batches execution so that the the average deviation between the predicted completion time and the times actual observed is minimized. SpeQuloS returns to user the predicted time associated to a statistical uncertainty as the success rate (with a $\pm$ 20% tolerance) of predictions performed against previous batch executions stored in the history, as well as the amount of credits that should be ordered to guarantee the prediction validity.

5.1.1 Cloud Deployment Strategies

Several strategies are considered to decide when to start Cloud workers, how many of them to use according to the amount of Credits supplied by an user, and how to use them.

Next, are three strategies used to decide when to launch Cloud workers:

- Completion Threshold (`9C`): Cloud workers are started when 90% of batch tasks are completed.
- Assignment Threshold (`9A`): Cloud workers are started when 90% of batch tasks have been assigned to DG workers.
- Execution Variance (`D`): Cloud workers are started when tasks execution time increases: Let $t_c(x)$ be the time at which x percent of batch jobs are completed and $t_a(x)$ be the time at which x percent of batch jobs were assigned to workers. The execution variance is $var(x) = t_c(x) - t_a(x)$. A sudden increase of execution variance denotes the beginning of the Tail effect. When execution variance reaches 2 times the maximum execution variance measured during the first half of the batch execution, the Cloud worker are started. The rational behind Execution Variance is to avoid provisioning Cloud resources for BoT where the tail do not appear.

If users allocate credits equivalent to S cpu.hours of Cloud usage, two approaches on how many Cloud workers to start are considered:

- Greedy (`G`): S workers are started. If a Cloud worker does not receive a task to compute, it stops to don't waste credit and let other Cloud workers to complete their tasks.
- Conservative (`C`): At any time t_e, an estimated remaining time to complete a batch t_r can be computed thanks to the Information module as follows:

$$t_r = t_c(1) - t_e = t_c(1) - t_c(x_e) = \frac{t_c(x_t)}{x_t} - t_c(x_t)$$

  where $t_c(x)$ is the elapsed time when x percent of batch tasks is completed ($\frac{t_c(x)}{x}$ is the batch completion rate). Conservative strategy starts $\max(\frac{S}{t_r}, S)$ Cloud workers. Therefore, there will be enough credits to run them until the estimated time remaining to complete the batch.

We propose three strategies to deploy and use the Cloud resources:

- Flat (`F`): Cloud workers behave as any regular workers, and the DG do not make distinction between them. Cloud and regular worker are in competition to execute tasks in this strategy.
- Reschedule (`R`): Cloud workers and regular workers are differentiated by DG server: When Cloud workers request for tasks, if all uncompleted tasks are being executed by regular workers, the DG server creates new duplicates of these tasks to assign them to Cloud workers. Therefore, tasks executed on regular workers that may cause the Tail effect are rescheduled on Cloud resources.
- Cloud Duplication (`D`): Cloud workers and regular workers are totally separated. When Cloud resources are provisioned, a new DG server is created in the Cloud and all uncompleted tasks from the original DG server are copied to the Cloud server. Then Cloud workers process tasks from this server and the results are merged back inside the original server, allowing the tasks execution by Cloud resource during Tail effect.

Implementation of these strategies differs in terms of complexity. While Flat does not need any particular DG modification, Reschedule requires to patch the DG server in order to adapt the scheduling process, and Cloud Duplication needs to implement task duplication and results merging between original and Cloud DG servers. The Flat strategies correspond to the solution proposed in the first prototype of SpeQuloS developed in the first year of the project, while Reschedule and Cloud Duplication are new proposition.

5.2 QoS Scheduler

5.2.1 Cloud Workers Management

The Scheduler module handle management of Cloud resources. When credits are allocated to support a batch execution, and according to strategies on when and how provision Cloud support, the Scheduler module will remotely start, configure and stop the Cloud workers. Scheduler operations to manage QoS support for batches and Cloud workers are presented in algorithms 1 and 2.

5.3 Performance Evaluation

This section presents SpeQuloS performance in providing QoS for batches executed in Desktop Grids. To obtain synthetic traces of batch executions, we have developed simulators of BOINC and XWHEP, as well as a simulator of the SpeQuloS framework.

Algorithm 1 Monitoring Batches

```
for all B in CreditSystem.getBatchesWithQoSOrder() do
  if not (isSupportedByCloudWorkers(B)) then
    if Oracle.shouldUseCloud(B) then
      if CreditSystem.hasCredits(B) then
        configure_QoS(B.getDG())
        for all CW in Oracle.cloudWorkersToStart(B) do
          CW.start()
        end for
      end if
    end if
  end if
end for
```

Algorithm 2 Monitoring Cloud workers

```
for all CW in startedCloudWorkers do
  B ← CW.getSupportedBatch()
  if (Oracle.shouldStopCloud(B)) or (not CreditSystem.hasCredits(B)) then
    CW.stop()
  else
    CreditSystem.bill(B,CW)
  end if
  if not (isSupportedByCloudWorkers(B)) then
    CreditSystem.pay(B)
  end if
end for
```

Table 2: Properties of batches: *size* is the number of jobs, *nops/job* is the number of instructions per job and *arrival* is the repartition function of jobs submission time. *weib* is the Weibull distribution and *norm*, the Normal distribution.

	Size	**nops / task**	**Arrival time**
`SMALL`	1000	3600000	0
`BIG`	10000	60000	0
`RANDOM`	$norm(\mu = 1000, \sigma^2 = 200)$	$norm(\mu = 60000, \sigma^2 = 10000)$	$weib(\lambda = 91.98, k = 0.57)$

5.3.1 Evaluation Setup

Many studies addressed Desktop Grid nodes volatility [11, 10, 13, 12]. In particular the Failure Trace Archive [14] provides several data-sets with nodes availability of popular DG servers. We used the public volunteer computing project SETI@Home (`seti`) ran by BOINC[15], and the private Desktop Grid deployments at University Notre Dame, ran by Condor (`nd`) to study SpeQuloS performances. We also used actual traces from the XtremWebHEP part of the infrastructure [16, 17, 18].

Bag of tasks applications are a major source of DCIs workload. We used the definition of a BoT given in [19, 20]: A batch is an ordered set of n independent jobs: $\beta = \{J_1, ..., J_n\}$. All jobs in β have the same owner and the same group name or group identifier and refer to the same application.

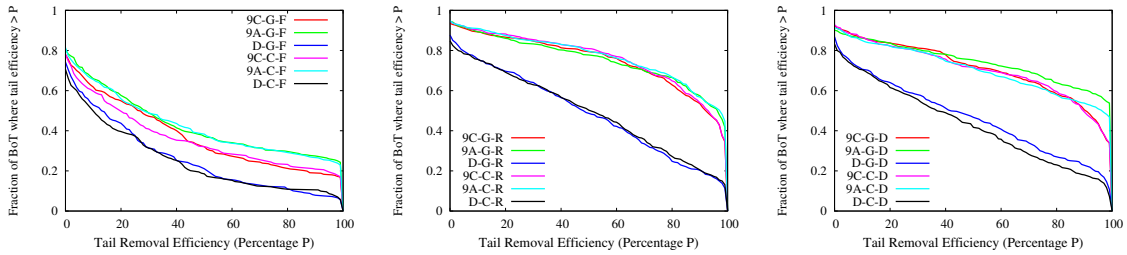
Jobs may not have been be submitted at the same time and may have a variable number of operations (*nops*) per jobs. BoT are also defined by there *size* i.e. the number of jobs. We used 3 BoT categories in our experimentation, denoted by `SMALL`, `RANDOM` and `BIG`. Those batches vary by size, jobs number of instructions and submission times. Table 2 summarizes the batch properties, observed from the EDGI infrastructure and previous studies [19]. As shown in the table, `SMALL` and `LARGE` BoT are homogeneous, whereas `RANDOM` is heterogeneous, with statistically generated properties.

The simulator uses pseudorandom number generator seed value to ensure reproductability of simulation executions which a allows fair comparisons between BoT executed with or without SpeQuloS. SpeQuloS users choose how many credits they order to enable QoS in their BoT. In our simulations, we set the number of credits ordered to be equivalent to 10% of the total batch workload in Cloud *CPU.hour*. Simulators execute BoT which are described in table 2 on Desktop Grids `seti` and `nd`), using both BOINC and XWHEP middleware. Various batch submission times were used along the node availability traces to consider several execution conditions. More than 25000 BoT executions were simulated to produce the results presented in the section. We have used EDGI and XtremWeb-HEP to lead the simulation campaign the French national Grid: Grid5000. This represents thousands of CPU hours and during the first semester of 2011, we have been amongst the largest users of Grid5000.

5.3.2 QoS Strategies Evaluation

We investigate the performance of SpeQuloS Cloud deployment strategies introduced in Section 5.2. Every combination of the strategies is considered to figure out which one lead to the best performances. The strategy combinations is named as follows: `9A-G-D` means that Cloud workers will start when 90% of the tasks have been assigned (Assignment Threshold), all the Cloud workers are started at once (Greedy) and the tasks which belong to the tail are all duplicated to the Cloud (Cloud Duplication).

Our first experiment compares the efficiency of the Cloud deployment strategies to remove



(a) Flat deployment strategy (b) Reschedule deployment strategy (c) Cloud duplication deployment strategy

Figure 4: Complementary cumulative distribution functions of Tail Removal Efficiency for several combinations of Cloud deployment strategies. Tail removal efficiency denotes the reduction percentage of the tail duration using SpeQuloS compared to without SpeQuloS.

the tail effect. *Tail Removal Efficiency (TRE)* is defined as the percentage reduction of the tail duration when SpeQuloS is used, compared to situation where SpeQuloS is not used. TRE is calculated as

$$TRE = 1 - \frac{t_{speq} - t_{ideal}}{t_{nospeq} - t_{ideal}}$$

where $t_{nospeqs}$ is the completion time measured without SpeQuloS (which is likely to be affected by tail), t_{speq} is the completion time measured for the same batch execution when SpeQuloS is used and t_{ideal} is the ideal completion time that could be obtained with that execution without tail effect.

Figures 4a, 4b and 4c present the complementary cumulative distribution function of TRE, using all combinations of Cloud deployment strategies. For a given efficiency, the figures show the fraction of batch executions which obtained a greater efficiency.

Figures show that all strategies can reduce tail effect. Best cases (Fig. 4c, `9A-G-D, 9A-C-D`), show that tail disappear in 50% of batch executions (TRE=100%), and that in 80% of batch executions, the tail is reduced by half.

Comparing strategies shows that the Flat strategy is worse than the others, with half of executions showing an efficiency not higher than 30%. Reschedule and Cloud Duplication strategies perform better in up to 80% of cases if the Execution Variance is excluded. Figures 4b and 4c also show that any Completion threshold or Assignment threshold strategies combined with Greedy or Conservative strategies give the best efficiency. The Assignment threshold strategy performs slightly better than the Completion threshold strategy, and Reschedule performs slightly better than Cloud duplication, in particular if Completion threshold is used.

The Flat strategy performance is lower than the others because Cloud workers are in competition with regular ones. With Flat strategy, jobs are assigned to Cloud workers without distinguishing them from regular workers. This leads Cloud workers to not receive jobs from the server during the tail part of batch execution. The Execution Variance strategy is also less efficient than the others. We observed that this strategy often detects the tail too late and therefore Cloud workers are not started in time to reduce tail effect significantly.

5.3.3 Cloud Resource Consumption

Another metric of strategies performance is the usage of the Cloud resources. A strategy with lower Cloud usage is better. SpeQuloS CreditSystem bills 15 credits to users when 1 CPU.hour of Cloud workers is used[1]. Therefore, the Cloud usage is related to the number of credits spent during the SpeQuloS execution to support BoT and can be used as a metric.

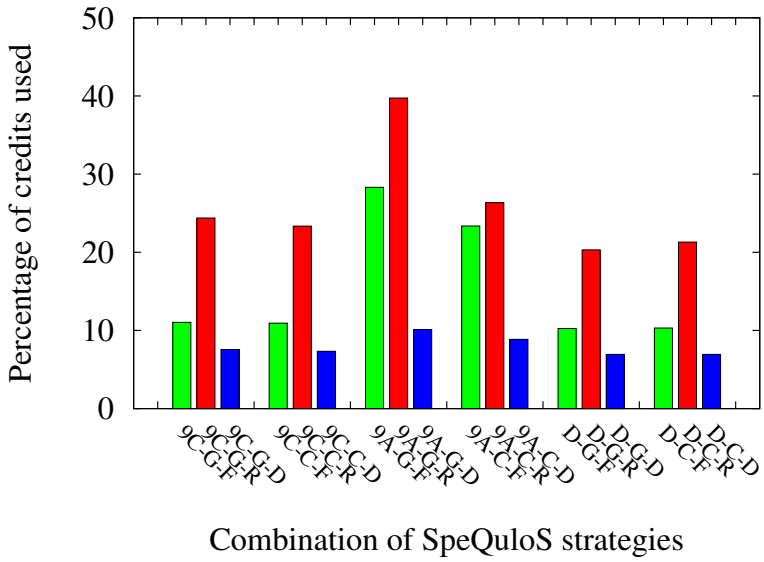


Figure 5: Credits consumption of various SpeQuloS strategies combinations. Lower is better.

Figure 5 presents the average percentage of credits provisioned that have actually been spent. In most of cases, less than 25% of credits are spent. As we provisioned in our simulation an equivalent of 10% of the total batch workload in terms of Cloud worker *CPU.hours*, these results show that actually, less than 2.5% of a batch workload is executed by Cloud workers. The figure also shows that Cloud duplication strategy has a lower consumption than Flat, which has a lower consumption than Reschedule. Indeed, Cloud workers are continuously busy in this strategy because they receive duplicates of uncompleted job until the batch is totally completed. Unsurprisingly, the results show that Assignment threshold have a slightly higher consumption, because it starts Cloud workers earlier, and that Conservative method saves more credits than Greedy.

In any cases, the strategies have a low credit consumption. This allows batch execution to be supported by Cloud resources up to its completion, and to let more credits available to other users. To give an idea of SpeQuloS applicability, if we consider that 20% of the BoT executed on EDGI asks for QoS, and EDGI would dispose of 250 Cloud nodes, then SpeQuloS could support up to 50000 active DG workers, which is close to the size of SETI@Home.

5.3.4 Completion Time Speed Up

Figures 6a, 6b and 6c present the average batch completion times observed when SpeQuloS is used or not. Results presented use Completion threshold, Conservative and Reschedule (9C-C-R)

[1]This value was arbitrarly choosen at the beginning of the project when SpeQuloS was running once every 4 minutes, thus 15 times per hour. Later, the value was kept because it has been found convenient to use.

strategies, which have shown to be efficient in removing Tail Effect, while keeping a low credits consumption, measured to be less than 2.5% of the total batch workload in equivalent Cloud's *cpu.h*.

Figures show results from BOINC and XWHEP middleware and for SETI@HOME (SETI) and NotreDame (ND) Desktop Grids, for which availability traces have been collected from the Failure Trace Archive ([14]). Figures also present result obtained with various batch workload: "BIG", which is a large batch of small tasks ; "SMALL", which is a small batch of long tasks, and "RANDOM", which is a heterogeneous batch, with a variable number of tasks of different length.

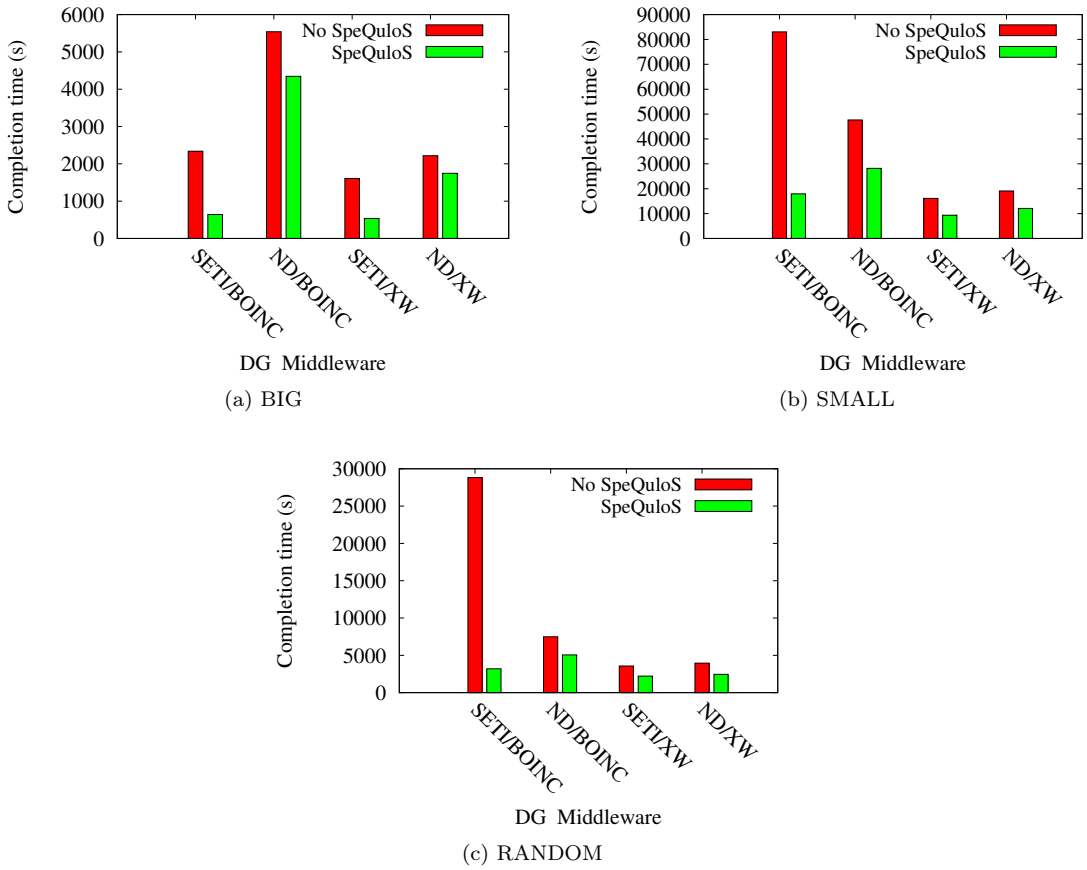


Figure 6: Average completion time measured with and without SpeQuloS for BOINC and XWHEP middleware in various Desktop Grids.

In any case, SpeQuloS is able to decrease batch completion time. Depending on middleware and DG considered, SpeQuloS speeds up the execution from 1.5 to 9 times. Best results are observed with BOINC and SETI, which includes highly volatile resources. As the tail effect is stronger here, SpeQuloS greatly improve the performance.

Results also depend on the batch workload. batches which are made of long tasks (`SMALL`) or heterogeneous tasks (`RANDOM`) are more likely to see large improvement. Indeed, without SpeQuloS, it is more difficult to execute this kind of batches on Desktop Grid, because low power and high volatility characteristics of their nodes.

Table 3: SpeQuloS completion time prediction percentage of success. Results obtained for various DGs, middleware and batch workloads are reported. A successful prediction means that the batch completion time is comprised between ± 20% of the predicted completion time.

	Batch category & Middleware						
	SMALL		BIG		RANDOM		
BE-DCI	BOINC	XWHEP	BOINC	XWHEP	BOINC	XWHEP	**Mixed**
seti	100	100	100	82.8	100	87.0	95
nd	100	100	100	100	100	96.0	99.4
Mixed	100	100	100	91.4	100	91.5	**97.5**

BOINC and XWHEP behave differently: BOINC implements tasks replication whereas XWHEP does not, and they also handle node failures in a different fashion. Thus, they cannot be compared, but it can be noted that using SpeQuloS leads to higher performance improvement in BOINC than in XWHEP.

5.3.5 Prediction Accuracy

Results presented use Completion threshold, Conservative and Reschedule (9C-C-R) strategies, which have shown to be efficient in removing Tail Effect, while keeping a low credits consumption, measured to be less than 2.5% of the total batch workload in equivalent Cloud's *cpu.h*.

SpeQuloS prediction results are presented in table 3. Prediction mechanism, described in Section 5.1, are made at 50% of batch completion. A prediction is reported to be successful when the completion time actually observed is comprised between ± 20% of the predicted time.

The α adjustment factor is computed using a "learning phase" of 30 executions using the same DG availability trace, middleware, and batch workload. For the results presented here, the learning phase adjusted α from 1 to a value comprised between 0.85 and 1.45, depending on the combination of DG traces, batch workload and middleware considered.

SpeQuloS prediction success rate is high: It is 97.5% when considering every results mixed. This means that in 97.5% of cases, the SpeQuloS is valid within an uncertainty of ± 20%. This results are considered very satisfactory considering that Desktop Grids are composed of volatile and heterogeneous resources and that without SpeQuloS there exist no solution to accurately predict BoT execution time.

Heterogeneous batches (RANDOM) give lower prediction success rate. As task sizes in such batch can highly vary, this observation is not surprising as prediction cannot completely rely on past executions.

5.4 Conclusion on Performances

These results have highlighted the effectiveness of combining Cloud resources to Desktop Grids in order to enhance QoS of the EDGI infrastructures. Our results show that QoS is greatly enhanced, for every DGs availability traces and BoT workloads we studied. The major results of our evaluation show that:

- SpeQuloS addresses the tail effet, which is the major cause of QoS problem with BoT executed on DGs. The tail is suppressed in half of cases, and in the other half it is significantly reduced.
- SpeQuloS only requires few Cloud resources to be efficient. We observed than an average of 2.5% of a BoT workload is offloaded on the Cloud.

- SpeQuloS greatly speeds up BoT completion time. According to the strategies that we have proposed, we observed that the speedup is higher than 2 in most of the cases, up to 9 in best cases.
- SpeQuloS delivers accurate batch completion time predictions to users, with a success rate of 97.5% on average.

6 Task JRA2.3: QoS Support In Science Gateway

6.1 Credit System

One of EDGI JRA2 goal is to provide a framework for a Credit system for EDGI users to use QoS features. The Credit system as several objectives:

- Cloud usage billing and accounting, to ensure that a single user does not improperly use all Cloud resources for himself.
- To build a framework to fairly share Cloud resources between Institutions that use the EDGI infrastructure.
- To encourage providing computing resources to EDGI Desktop Grids by allocating some QoS credits to volunteer participants.
- To encourage users to use QoS while avoiding the waste of Cloud resources.

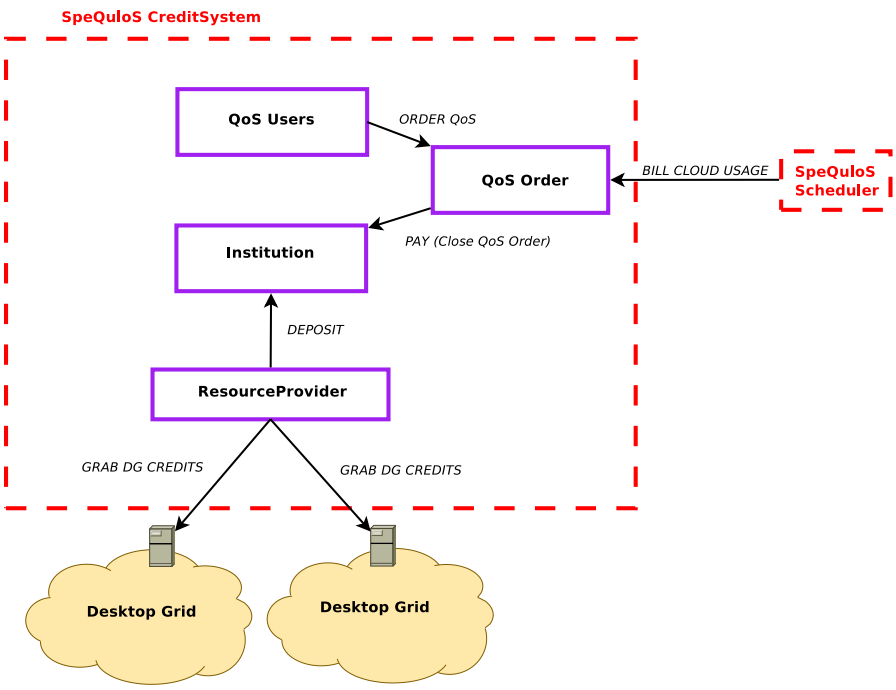


Figure 7: SpeQuloS Credit System entities

SpeQuloS includes such a Credit system, implemented in the Credit System module. Its role is handle all credits related operations. SpeQuloS Credit System introduces the following entities, as summarized in figure 7:

- QoS Users: The individual EDGI users of the SpeQuloS QoS features.
- QoS Orders: QoS requests created by QoS users to support one of their batch execution.
- Institutions: The structural organization grouping QoS Users, such as a laboratory, a university, a scientific community. Each Institution holds an account with "QoS credits" that is shared between its QoS users.

- Desktop Grid: A Desktop Grid (one server) belonging to EDGI infrastructure. The Desktop Grid individual resource is a worker node. Each worker nodes computing effort, related to the number and size of jobs they compute, is equivalent to a value called "DG credits".
- Resource Provider: A grouping of a Desktop Grid worker nodes that support the same institution. Worker nodes belonging to the same Resource Provider share the "DG credits" they won. Each Resource Provider choose an Institution to support. The Resource Provider and its supported Institution should share some affinity.

Here are some imaginary examples of the Credit System usage in the context of EDGI infrastructure:

- The CNRS Institution uses the EDGI infrastructure to provide computing resources to high energy physics researchers. CNRS decides to connect their laboratory desktop computers to XW @ LAL Desktop Grid as worker nodes. Worker nodes provided by the CNRS group themselve as the "CNRS @ LAL" Resource Provider. This Resource Provider declares to support the CNRS Institution to the SpeQuloS Credit System. Thanks to this participation to EDGI DGs, CNRS researchers will receive more QoS credits to improve performances of their batch executions.
- The INRIA Institution does not use the EDGI infrastructure but has a lot of unused computers. INRIA decides to connect them to XW @ LAL Desktop Grid as worker nodes. The INRIA worker nodes join the already existing "CNRS @ LAL" Resource Provider because they know that this Resource Provider supports CNRS, and they feel concerned by research done by this Institution.

6.1.1 Credit System Cloud Resources Billing

Using Cloud resources is costly, and with SpeQuloS, many QoS Users may want to use them to support theirs tasks execution. The SpeQuloS Credit System module implements Cloud usage accounting, billing and arbitration by providing an interface similar to a simple banking system with "order", "bill" and "pay" operations with some virtual QoS credits. The Credit System uses a fixed exchange rate, with 1 *CPU.hour* of Cloud worker is equivalent to 15 QoS credits. QoS Users are registered and they belong to an Institution, which hold an account with credits.

According to its QoS requirements, a QoS User must ask to the Credit System to "order" some QoS with the number of credits related to the amount of Cloud resources he needs. The order is accepted by the Credit System if there is enough QoS credits on the user Institution account. During batch execution, started Cloud workers are "billed" by Scheduler module to the Credit System. At the end of execution, if all credits have not been spent, they are transferred back to user's Institution. On the opposite, if all credits order are consumed before the end of batch execution, the Scheduler module stops using Cloud resources. In any case, the Scheduler call the Credit System's "pay" operation to close the order.

This system ensures that usage of Cloud resource is accounted and controlled. Users usage of Cloud is limited by the amount of QoS credits of their Institution.

6.1.2 Credit System Earning Policy

The other main function of the Credit system is to fund institutions, according to computational power offered to Desktop Grids, with credits that users can spend in Cloud resources to enable QoS for their batches. SpeQuloS proposes a policy to earn QoS credits with the following objectives in mind:

- To fund periodically Institution accounts to allow their QoS Users to order QoS for their batches.
- To give incentive to provide computing resources to DGs.
- To ensure that credits provided can really be spent regarding to Cloud resources available in the infrastructure.
- To leave some credits to QoS Users that do not participate in DGs so they can try QoS.

SpeQuloS Credit System periodically transfers some credits to Institutions accounts using a "deposit" function. This function convert DG credits earned by Resource Providers to QoS credits for Institutions. The DG credits earned by a Resource Provider are representative of computing participation of all worker nodes that group themselves to form this Resource Provider. SpeQuloS uses the deposit function presented by Algorithm 3, with respect to objectives described previously. This function ensures that Institution is funded proportionally with the amount of DG credits earned by Resource Providers which support it. It also guarantees that an unsupported Institution will receive free credits equal to half of the credits earned by the Institution receiving the lowest support. Finally, it avoids over funding an Institution, which may lead to having more credits than available Cloud resources by setting a maximum to Institution accounts.

Algorithm 3 DEPOSIT FUNTION

```
CreditsPerDay ← #CloudCpu x CpuCostPerHour x 24
InstitutionAccountMax ← CreditPerDay x 7
for all R in ResourceProvider do
  DGCreditsForInstitution[R.getSupportedInstitution()] += R.getDGCreditsEarned()
  InstitutionMinimal ← min( InstitutionMinimal, DGCreditsForInstitution[R.getSupportedInstitution()])
end for
for all I in Institutions do
  if DGCreditsForInstitution[I] = 0 then
    DGCreditsForInstitution[I] ← 1/2 x InstitutionMinimal
  end if
  TotalDGCredits += DGCreditsForInstitution[I]
end for
for all I in Institutions do
  I.account += min( CreditsPerDay x DGCreditsForInstitution[I] / TotalDGCredits, InstitutionAccountMax)
end for
```

The Desktop Grid credits collected by Resource Providers of various Desktop Grids of the infrastructure are periodically monitored and stored by the Credit System using dedicated plugins, specific to each DG middleware. More detail on these plugins are given in section 7. If the Credit System discovers worker nodes that do not belong to a Resource Provider, it adds them to a virtual Resource Provider dedicated to this kind of nodes. If Resource Provider does not declare to SpeQuloS which Institution it supports, an arbitrary Institution is chosen by the Credit System.

6.2 QoS Portal

In the EDGI project, there is a web-based portal for the users to submit and control their jobs when utilizing the EDGI infrastructure. This portal is the WS-PGRADE/gUSE portal developed by SZTAKI. During the project this portal is used as an official one for the EDGI users.

6.2.1 About WS-PGRADE/gUSE portal

The WS-PGRADE/gUSE portal framework has been developed to support large variety of user communities. It provides a generic purpose high-level graphical user interface to create and run workflows on various DCIs including clusters, grids, desktop grids and clouds. The portal can be used by NGIs to support various small user communities who cannot afford to develop their own customized science gateway. The WS-PGRADE/gUSE portal framework also provides the ASM and Remote API interfaces to create application-specific science gateways according to the needs of different user communities.

WS-PGRADE portal is based on the gUSE (grid User Support Environment) service set and is the second generation P-GRADE portal that introduces many advanced features both at the workflow and architecture level compared to the first generation P-GRADE portal.

Details about how the portal is used to access the EDGI infrastructure can be found in Deliverable D3.1.

Details about the portal functionalities can be found at `http://www.guse.hu` .

6.2.2 Integrated functionalities

In the gUSE portal it is possible to utilize the services provided by SpeQuloS. The integration was done through the creation of web forms for the various SpeQuloS functions.

The functionalities provided by SpeQuloS are as follows:

- Ordering credits for QoS support: when the user decides to speed up a certain batch of jobs, credits must be allocated to make SpeQuloS adding extra resources for those jobs
- Get available credits: the user can query his/her currently available credits collected previously
- Get current QoS orders: the user can list his currently active orders submitted to SpeQuloS
- Get information on a batch completion: the user can query SpeQuloS to provide information on a certain batch about the elapsed time and completion rate
- Make prediction about batch completion time: the user can query how much credit should be spent in order to speed up a certain batch

In the next subsections, these five functionalities are detailed.

6.2.3 Ordering credits for QoS support

When the user decides to speed up a certain batch of jobs, credit can be allocated by using a form (see Figure 8) in gUSE. The form takes the following parameters:

- Batch ID: The batch identifier associated to the collection of jobs
- Desktop Grid ID: The identifier of the Desktop Grid where batch is executed
- User ID: The user identifier

- Credit: The number of credits to allocate to the order

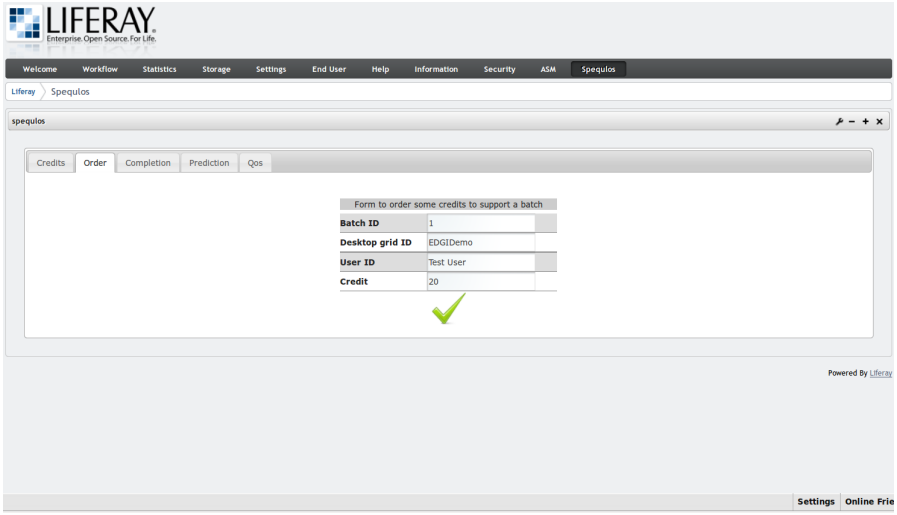


Figure 8: Ordering credits for QoS support in gUSE

When filling the field with the values for a certain batch, the request is sent to SpeQuloS and the response message is shown (see Figure 9) in a popup window.

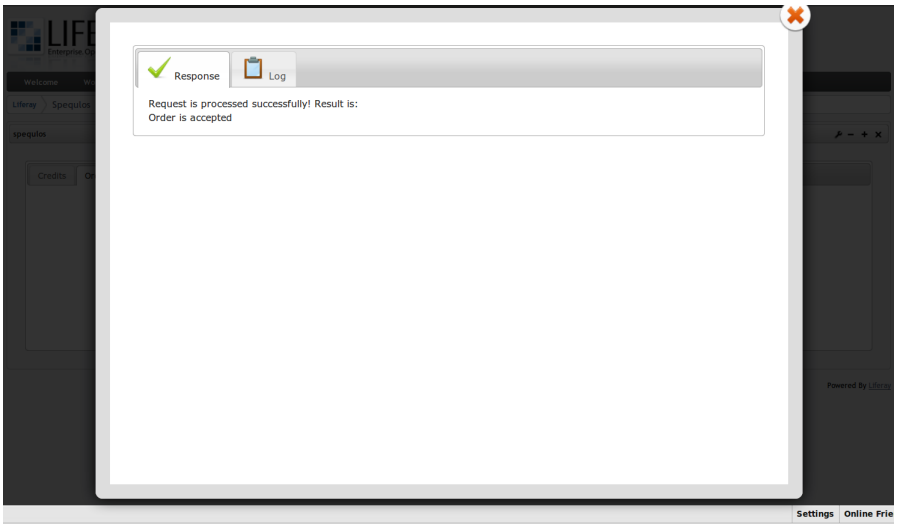


Figure 9: SpeQuloS response of ordering credits for QoS support in gUSE

On Figure 9) an accepted order is shown. During execution of the SpeQuloS command initiated by the user detailed logging messages are saved. Later the user can inspect and report to the administrator in case there are some internal errors. A sample log message window is shown on Figure 10.

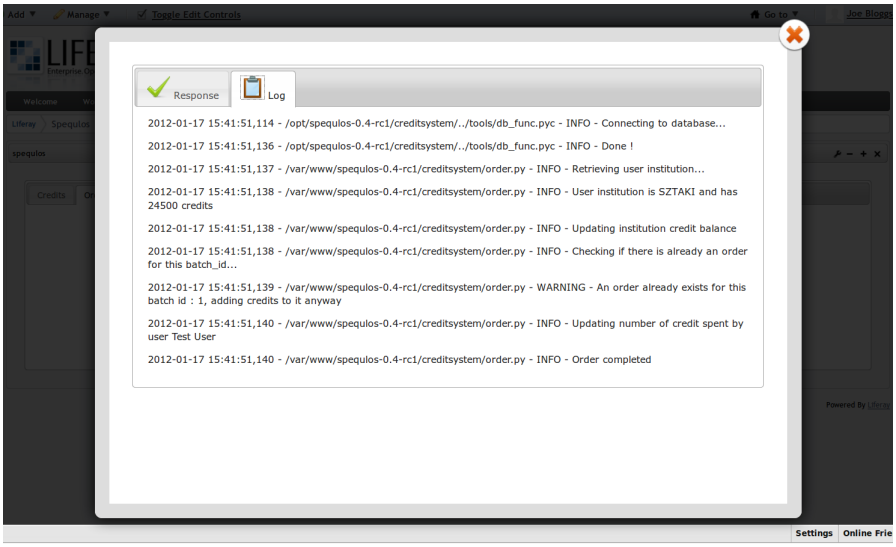


Figure 10: SpeQuloS log messages of ordering credits for QoS support in gUSE

6.2.4 Get available credits

The user can query the available credits in an institution account and available to him/her. For the query, the following fields must be defined:

- Batch ID: The batch identifier associated to the collection of jobs
- Desktop Grid ID: The identifier of the Desktop Grid where batch is executed
- Institution ID: The identifier of the institution the user belongs to
- User ID: The user identifier

The form associated to this query can be seen on Figure 11.

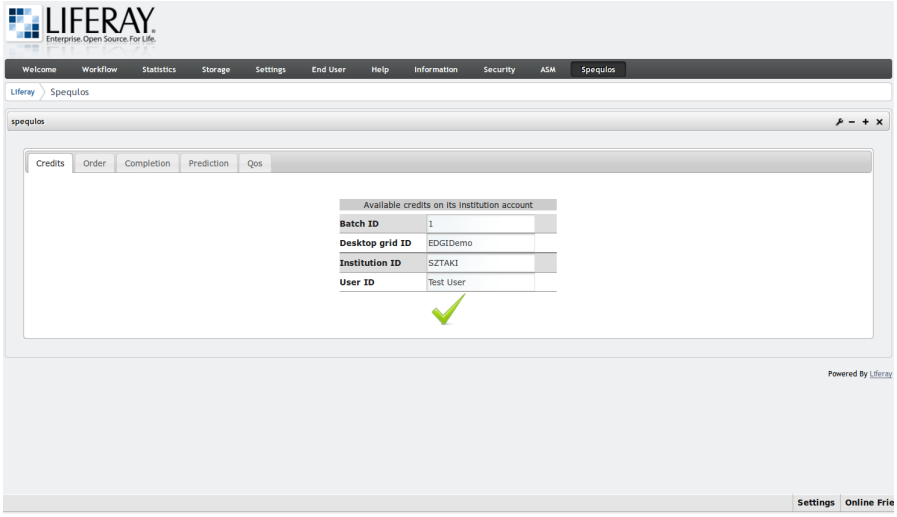


Figure 11: Query available credits from SpeQuloS in gUSE.

As a response the number of credits is displayed in a popup window similarly to the first function.

6.2.5 Get current QoS orders

The user can list his/her currently active QoS orders requested before. To do that the following fields must be filled:

- User ID: The user identifier

The form associated to this query can be seen on Figure 12.

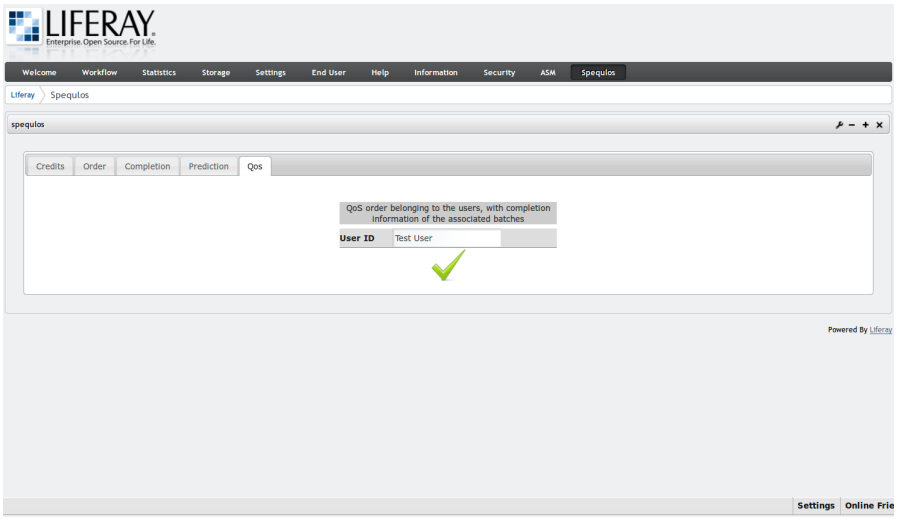


Figure 12: Query current QoS orders from SpeQuloS in gUSE.

On Figure 13 an example popup window can be seen showing the response message from SpeQuloS. In this window 3 different active QoS request is listed.

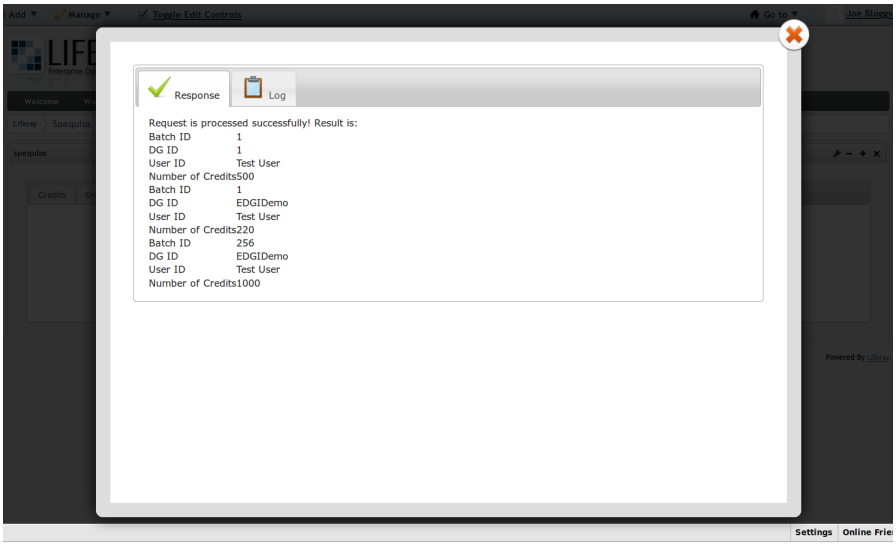


Figure 13: Response for query current QoS orders from SpeQuloS in gUSE.

6.2.6 Get information on a batch completion

The user can get information on a batch completion in gUSE. In order to do that the following information must be provided:

- Batch ID: The batch identifier associated to the collection of jobs
- Desktop Grid ID: The identifier of the Desktop Grid where batch is executed

The form associated to this query can be seen on Figure 14.

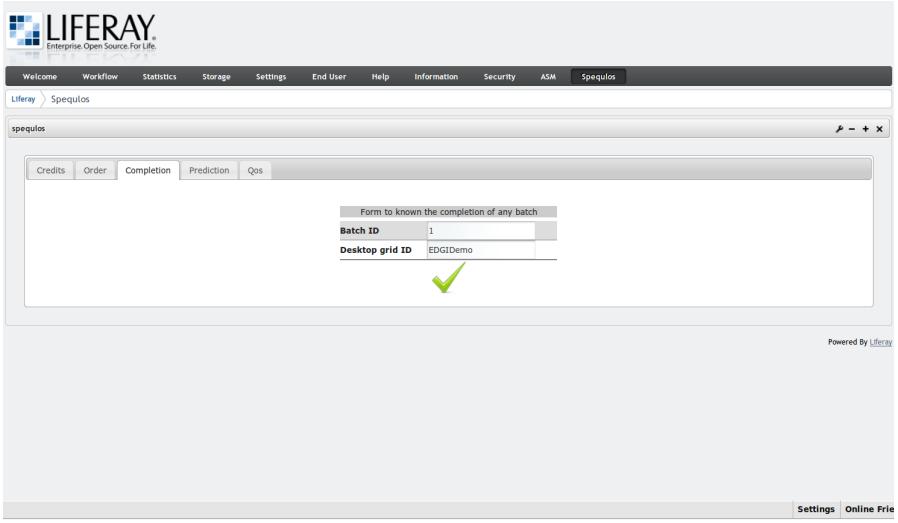


Figure 14: Get information on a batch completion in gUSE.

As a response to this query, the portal informs the user about

- the elapsed time since the batch submission and
- the complation rate in percent.

6.2.7 Make prediction about batch completion time

The user can query the completion time predicted by SpeQuloS for a certain batch of jobs in gUSE. In order to do that the following information must be provided:

- Batch ID: The batch identifier associated to the collection of jobs
- Desktop Grid ID: The identifier of the Desktop Grid where batch is executed

The form associated to this query can be seen on Figure 15.

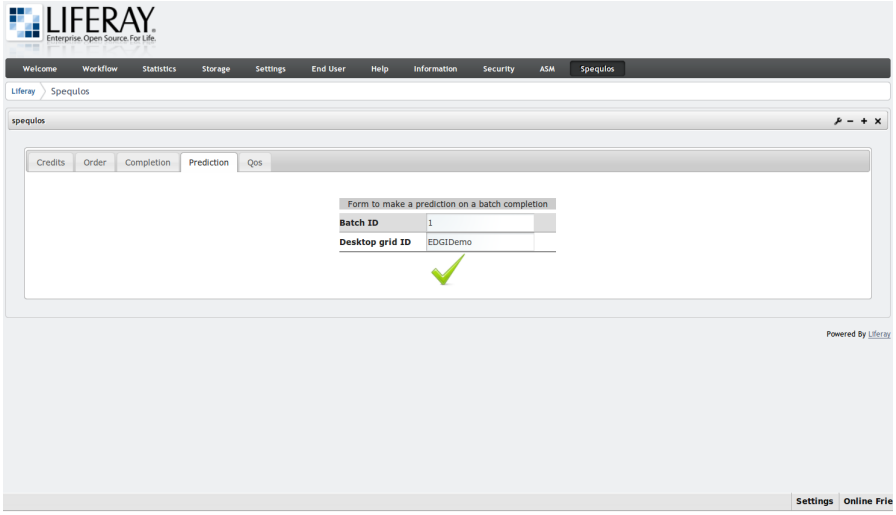


Figure 15: Make prediction about batch completion time in gUSE.

6.2.8 Availability

In the EDGI project, a WS-PGRADE/gUSE web-based scientific portal is maintained and publicly accessible. The forms introduced above can be found as part of this portal. The portal webpage is as follows: `https://edges-portal.sztaki.hu/`

7 New Software Development

This section reports on major developments inside SpeQuloS since the version delivered in the first year of the EDGI project, and reported in the deliverable D7.1.

The SpeQuloS version delivered by JRA2 is labeled 0.4. The first release candidate of SpeQuloS 0.4 has been publish on 13, January 2012. We plan to publish the final version once SA2 testing process has completed.

SpeQuloS uses the Python programming language, backed by MySQL databases to store information. SpeQuloS internal and external communications use Web Services. SpeQuloS does not use permanent daemon to run, but only requires some programs to be periodically executed using cron. Several DGs and Cloud services can be connected at the same time to a single SpeQuloS deployment. The SpeQuloS source code is publicly available (see A.1)

Here are the major changes inside SpeQuloS internals since the version delivered for EDGI's first year.

- Scheduler advanced strategies: The QoS strategies presented in section 5.2 has been implemented in SpeQuloS Scheduler and Oracle modules. The "90th percent of completion" (9C) and "Conservative" (C) methods have been selected to be implemented in Oracle's "should_start_cloud" and "how_many_cloud" functions. The Reschedule and Cloud Duplication strategies have been implemented for BOINC and XWHEP, respectively, as alternative to Flat strategy that was used before. SpeQuloS administrators can choose between Flat and the advanced strategy using a configuration option. More information and these implementations is given further in this section.

- Oracle prediction: The Oracle prediction method described in section 5.1 has been implemented as a replacement of "Calculate Cloud Benefit" method previously used.

- the Credit System earning policy: The Credit System operation, described in section 6.1, has been implemented. Most of new developments were related to the earning policy. The "credit_grab" function is used to grab credits earned by Resource Provider (RP) of each DG, using "plugins" specific to BOINC or XHWEP. The "deposit" function is used to convert DG credits earned by RP to SpeQuloS credits and to transfer them to Institutions accounts. The "admin_resource_provider" page is provided to let the SpeQuloS administrator select which Institution is supported by each RP.

- Parallelization of repetitive tasks: The code involving repetitive tasks (grabbing information from DGs, starting or stopping Cloud workers) has been modified to be executed in parallel instead of sequence, using multithreading.

- Administration web pages: SpeQuloS can be configured through dedicated administration pages, using web services. These pages ease the registration of DGs, Cloud Services, Institutions, Users, etc. to SpeQuloS. Web services allow to perform these tasks remotely, or to interconnect SpeQuloS configuration with external components.

The rest of the section will present other specific new features developed in SpeQuloS.

7.1 Cloud Workers Management

SpeQuloS used to manage Cloud worker starting and stopping using "libcloud" library. In addition to "libcloud", we added a new handler to manage Cloud workers, called "command". This handler allows to use any command line to start or stop Cloud workers. This eases the use of the Cloud services which are not supported by libcloud in SpeQuloS, and allows to use any

customized tool provided by the users. In particular, the "command" handler is used inside the EDGI infrastructure to interconnect SpeQuloS with the 3G-Bridge.

To allow Cloud workers to be aware of their execution context when "command" handler is used, SpeQuloS sets up several variables to be available to the command executed, such as the Cloud service and Desktop Grid involved, or the Batch identifier which must be supported by the Cloud workers.

7.2 Supporting XWHEP

This part describes the developments specific to XWHEP support in SpeQuloS.

7.2.1 Support for Information Module

The Information module uses a dedicated plugin to grab information from a XWHEP DG. Therefore, it requires to install a PHP web page on the server. This plugin is used to fetch batch information from the XWHEP database. The plugin has been improved during last developments, in particular to be compatible with the new Oracle prediction method.

7.2.2 Support for Credit System Module

The Credit System module also uses a dedicated plugin to grab credit related to information from a XWHEP DG. A PHP web page is also used on the XWHEP server. This plugins did not exist in prior version, and has been created to support new Credit System earning policy.

As no credit system exists in XWHEP, the plugins cannot directly grab credits information from a XWHEP server. As an equivalency to credits, we use the execution time of workers nodes divided by a constant value. The membership of a worker to a specific Resource Provider is defined by the XWHEP "project" parameter. Workers with the same value of "project" parameter are considered to belong to the same Resource Provider. Workers which do not have the "project" parameter filled are also grouped together to form a “catch-all” Resource Provider.

7.2.3 Cloud Duplication Strategy

The Cloud Duplication strategy, described in section 5.2, is implemented for XtremWeb-HEP middleware. The principle of Cloud Duplication strategy is to duplicate uncompleted jobs to a new DG server (called the *Cloud server*), that will only serve Cloud workers. When Cloud workers complete jobs, results are copied back from Cloud server to original DG server. This strategy ensures that Cloud workers are not in competition with regular work and receive jobs to compute until all jobs are completed.

The Cloud duplication strategy requires a dedicated XWHEP server (the Cloud server). However, a single server is needed to handle Cloud duplication of several BoT, even from different original DGs. This is achieve using XWHEP “batchid“ option in the Cloud worker configuration file, which ensures that a worker will only fetch jobs belonging to the indicated batch. In addition, a XWHEP client must be installed on the server to manage it, as well as scripts to copy uncompleted job from original DG to Cloud server and to merge back jobs completed on Cloud server to DG server (respectively called "dg2cloud.sh" and "cloud2dg.sh"). These scripts use XWHEP clients to connect to the original DG server and the Cloud server. Their roles are to get and submit applications, jobs and results.

Both script are run each 15 minutes using cron, to ensure that jobs newly submitted to DG are copied to Cloud server, and that completed job on the Cloud server are quickly copied to DG. The algorithm implemented are presented by algorithms 4 and 5.

Algorithm 4 DG2CLOUD.SH PARAMETERS: BATCH-ID, DG-CLIENT, CLOUD-CLIENT

```
Jobs ← DG-CLIENT.getUncompletedJobsFromGroup(BATCH-ID)
if not (CLOUD-CLIENT.getGroups(BATCH-ID)) then
  CLOUD-CLIENT.createGroup(BATCH-ID)
end if
if not (CLOUD-CLIENT.getApps(Jobs[0].getApp())) then
  App ← DG-CLIENT.downloadApp(Jobs[0].getApp())
  CLOUD-CLIENT.uploadApp(App)
end if
for all J in Jobs do
  if not (CLOUD-CLIENT.getJobs(J)) then
    CLOUD-CLIENT.submitJob(J,App)
  end if
end for
```

Algorithm 5 CLOUD2DG.SH PARAMETERS: BATCH-ID, DG-CLIENT, CLOUD-CLIENT

```
Jobs ← CLOUD-CLIENT.getCompletedJobsFromGroup(BATCH-ID)
for all J in Jobs do
  if not (DG-CLIENT.isCompleted(J)) then
    R ← CLOUD-CLIENT.getResult(J)
    DG-CLIENT.uploadResult(R)
    DG-CLIENT.updateJob(J,status="COMPLETED",result=R)
  end if
end for
```

SpeQuloS can use the Cloud Duplication strategy to support BoT executed on XWHEP Desktop Grids. The Cloud Duplication strategy is triggered by the SpeQuloS scheduler when Cloud resources are used for each batch supported by QoS. The start of the Cloud Duplication is triggered at the "configure_QoS" step, i.e. just before the scheduler starts the Cloud workers to support a BoT for the first time. The Cloud duplication ends at the "unconfigure_QoS" step, i.e. when Cloud resources stop to support a BoT (because the BoT is completed or the provisioned credits are spent).

Whatever is the Cloud handler used (libcloud or command), the Cloud workers must be configured to ensure that Cloud workers will process jobs form Cloud server (and not from original DG server). A reference implementation, that must be adapted according to the infrastructure where SpeQuloS is deployed of the Cloud Duplication is provided in SpeQuloS package. More information on Cloud Duplication strategy in SpeQuloS can be found in A.2 section.

7.3 Supporting BOINC

This part will describe the developments specific to BOINC support in SpeQuloS.

7.3.1 Support for Information Module

The Information module uses a dedicated plugin to grab information from a BOINC DG. It requires to install a PHP web page on the server, as an additional BOINC "project administrator" pages.

7.3.2 Support for Credit System Module

The Credit System module also uses the BOINC statistic system to grab credits from a BOINC server. This does not requires to add anything to a BOINC server, but to enable the export of XML statistics in BOINC server configuration. On the SpeQuloS side, a plugin called by the Credit System grabbing function fetches information from statistics exported by the BOINC server.

The membership of a BOINC client (worker) to a specific Resource Provider is defined by the BOINC "team" parameter. Clients with the same value of the "team" parameter are considered to belong to the same Resource Provider by SpeQuloS. Client without the "team" parameter are also grouped together to form a “catch-all” Resource Provider.

7.3.3 Reschedule Strategy

The Reschedule strategy, described in section 5.2, is implemented for the BOINC middleware. The principle of the Reschedule strategy is to specifically assign uncompleted jobs to Cloud workers. When Cloud workers ask for tasks, the BOINC scheduler duplicates uncompleted jobs and assigns them to Cloud workers. This strategy ensures that remaining jobs are computed as soon as possible on the stable and reliable Cloud worker resources.

The Reschedule strategy requires a patch to the BOINC server. We provide this patch as well as complete documentation (see section A.2) to apply it. A PHP file, called "assign_batch_to_cloud" must be copied to BOINC "project administrator" page. This file is remotely called by SpeQuloS scheduler at "configure_QoS" steps, to trigger the reassignment of remaining jobs to a specific BOINC client, called Cloud Client, an identified by a specific identifier that can be set in BOINC project configuration. This identifier will be used for Cloud workers connection to server.

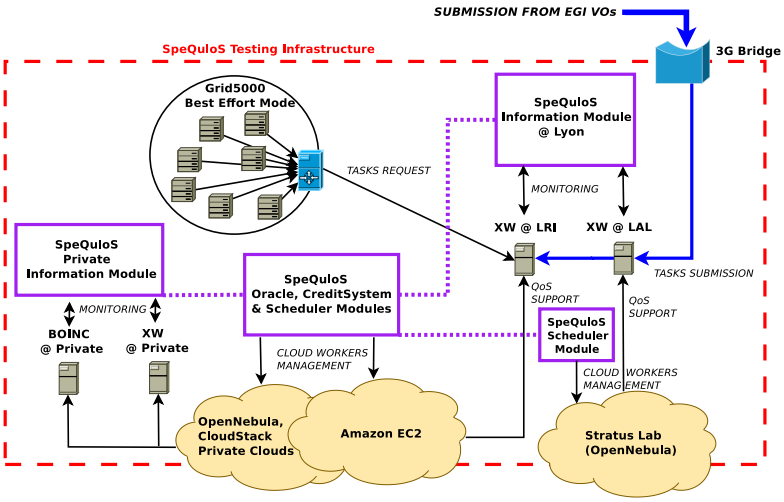


Figure 16: SpeQuloS deployment as a part of the EDGI Development infrastructure.

7.4 Deployment in Development Infrastructure

Figure 16 presents the current deployment of SpeQuloS inside its development infrastructure. This testing infrastructure is used to test permanently SpeQuloS new features and strategies before going into SA2 testing infrastructure and SA1 production infrastructure. SpeQuloS' modules are split and duplicated across the deployment.

Several DGs are used in SpeQuloS development infrastructure. We deployed our own private XWHEP and BOINC DGs but we also used the production DG deployed at Laboratoire de Recherche en Informatique that uses XWHEP middleware. To include computing resources that reproduce volatile characteristics of DG nodes, 6 clusters of Grid'5000 are used in best effort mode, meaning that they can be preempted at any time by other Grid'5000 users.

Amazon EC2 is used by SpeQuloS to support batch executed on DGs. Other Cloud service have been connected to the infrastructure, to test SpeQuloS compatibility with various Cloud technologies. Private Cloud services based on OpenNebula and CloudStack have been tested. The StratusLab infrastructure, which includes an OpenNebula Cloud service, has also be connected to SpeQuloS to support batches execution on the Laboratoire de l'Accélerateur Linéaire laboratory.

The framework has been tested against real-world applications from the EDGI application Repository. Applications running in the development infrastructure include DART, a Framework for Distributed Audio Analysis and Music Information Retrieval by Cardiff University, BNB-Grid, which aims at solving hard combinatorial, discrete and global optimization problems, and ISDEP, which is a fusion plasma application which simulates the Tokamak of ITER.

8 Conclusion

This document has presented JRA2 tasks "Advanced QoS Scheduler and Oracle" and "Support In Science Gateway". We introduced SpeQuloS, a framework dedicated to increase QoS delivered by Desktop Grids by supplying additional Cloud resources. SpeQuloS supervises BoTs executed by Desktop Grids, and supplies Cloud resources to execute the last fraction of the batch to mitigate the Tail effect, which harms execution performance in Desktop Grids. Here are the main achievements of JRA2 team described in this deliverable:

- Thanks to the collected by the JRA2.1 Information task, the tail effect has been characterized in terms of completion time slowdown and fraction of the tasks causing the tail.
- Several combination of strategies to deploy Cloud resources have been investigated and compared according their ability in removing the tail effect, as well as their Cloud resources consumption.
- Performance evaluation showed that SpeQuloS is able to significantly improve QoS in Desktop Grids by speeding up batch completion time and performing accurate execution time predictions.
- SpeQuloS Credit System allows Cloud usage billing to EDGI user as well as incentive to participate to the EDGI infrastructure by providing QoS credits.
- The QoS portal provides easy and integrated access to SpeQuloS features for EDGI users.
- SpeQuloS implementation has been updated to include th aforementioned items, and is ready to be tested and deployed by EDGI SA2 SA1 Work Packages.

SpeQuloS deployment inside the European Desktop Grid Infrastructure demonstrates the feasibility and relevance of hybrid infrastructures involving Desktop Grids and Clouds to address problems of actual high performance computing users.

8.1 Publications

The project has published papers [21, 22, 23, 24, 25] in the following venues:

- *Workers in the Clouds* Attila Csaba Marosi and Peter Kacsuk PDP 2011 - The 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing, Ayia Napa, Cyprus
- *Hybrid Distributed Computing Infrastructure Experiments in Grid5000: Supporting QoS in Desktop Grids with Cloud Resources* – S. Delamare, G. Fedak, D. Kondo and O. Lodygensky . – Grid500 Spring School (peer reviewed), 2011. **[Best presentation award]**
- *Towards Hybrid Desktop Grid and Cloud Infrastructures* – G. Fedak and S. Delamare – Book Chapter – Desktop Grid Computing – CRC Press, 2011.
- *SpeQuloS: A QoS Service for BoT Applications Using Best Effort Distributed Computing Infrastructures* – S. Delamare, G. Fedak, D. Kondo and O. Lodygensky . in International Symposium on High Performance Distributed Computing (HPDC'2012), Delft, Nederlands, 2012
- *SpeQuloS: A QoS Service for Hybrid and Elastic Computing Infrastructures.* S. Delamare, G. Fedak, D. Kondo, and O. Lodygensky. Journal of Cluster Computing (Cluster), SI: Selected paper from HPDC'12, 2013

8.2 Measures/indicators

One indicator to assess the QoS effort for this deliverable is the number of Clouds supported and the objective is set to two, as we currently support OpenNebula and OpenStack Cloud services used in EDGI infrastructure. However, several other Cloud services have been successfully tested with SpeQuloS, including Amazon EC2, Rackspace, StratusLab and Grid5000 as a Cloud.

Other indicator for this deliverable is the number of Desktop Grid technologies supported and the objective of two technologies is reached, as we currently support both BOINC and XtremWeb.

The indicator measuring the number of accepted publications is also reached as 2 publications were required.

A JRA2 D7.2 ressources

All the resources are also available in the EDGI project SVN.

A.1 SpeQuloS package

SpeQuloS 0.4-rc1 is available since 13rd January 2012. It can be downloaded at:
`http://graal.ens-lyon.fr/~sdelamar/spequlos/spequlos-0.4-rc1.tar.gz`

A.2 SpeQuloS documentation

A.2.1 SpeQuloS installation documentation

The installation documentation is provided in the SpeQuloS package. It can be found in the file "install.txt", under the "docs" subdirectory of the archive. It can also be found at the url:
`http://graal.ens-lyon.fr/~sdelamar/spequlos/install.txt`

A.2.2 SpeQuloS usage documentation

The user documentation is provided in the SpeQuloS package. It can be found in the file "usage.txt", under the "docs" subdirectory of the archive. It can also be found at the url:
`http://graal.ens-lyon.fr/~sdelamar/spequlos/usage.txt`

A.2.3 Cloud Worker image creation documentation

The documentation describing how to build a Cloud Worker image to be used by a Cloud service is provided in the SpeQuloS package. It can be found in the file "how_to_create_a_cw_vm.txt", under the "docs" subdirectory of the archive. It can also be found at the url:
`http://graal.ens-lyon.fr/~sdelamar/spequlos/how_to_create_a_cw_vm.txt`

A.2.4 BOINC server patching documentation

The documentation describing how to patch a BOINC server to be supported by SpeQuloS is provided in the SpeQuloS package. It can be found in the file "install_boinc_for_spequlos.txt", under the "docs" subdirectory of the archive. It can also be found at the url:
`http://graal.ens-lyon.fr/~sdelamar/spequlos/install_boinc_for_spequlos.txt`

A.2.5 Cloud Duplication for XWHEP

The documentation describing how to deploy the Cloud Duplication for XWHEP servers to be used by SpeQuloS is provided in the SpeQuloS package. It can be found in the file "duplication.txt", under the "docs" subdirectory of the archive. It can also be found at the url:
`http://graal.ens-lyon.fr/~sdelamar/spequlos/duplication.txt`

A.3 QoS Portal resources

WS-PGRADE/gUSE portal Homepage:
`http://www.guse.hu`

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*, 2009.

[2] P. Kacsuk, Z. Farkas, and G. Fedak. Towards Making BOINC and EGEE Interoperable. In *Proceedings of 4th IEEE International Conference on e-Science (e-Science 2008), International Grid Interoperability and Interoperation Workshop 2008 (IGIIW 2008)*, pages 478–484, Indianapolis, USA, December 2008.

[3] Zoltan Balaton, Zoltan Farkas, Gabor Gombas, Peter Kacsuk, Robert Lovas, Attila Csaba Marosi, Ad Emmen, Gabor Terstyanszky, Tamas Kiss, Ian Kelley, Ian Taylor, Oleg Lodygensky, Miguel Cardenas-Montes, Gilles Fedak, and Filipe Araujo. EDGeS: the Common Boundary Between Service and Desktop Grids. *Parallel Processing Letters*, 18(3):433–453, September 2008.

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

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

[6] Gilles Fedak, Cecile Germain, Vincent Neri, and Franck Cappello. XtremWeb: A Generic Global Computing Platform. In *CCGRID'2001 Special Session Global Computing on Personal Devices*, 2001.

[7] 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.

[8] 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.

[9] 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.

[10] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien, and Henri Casanova. Resource Availability in Enterprise Desktop Grids. *Future Generation Computer Systems*, 23(7):888–903, August 2007.

[11] Derrick Kondo, Felipe Araujo, Paul Malecot, Patricio Domingues, Luis M. Silva, Gilles Fedak, and Franck Cappello. Characterizing Result Errors in Internet Desktop Grids. In *European Conference on Parallel and Distributed Computing EuroPar'07*, Rennes, France, August 2007.

[12] Paul Malécot, Derrick Kondo, and Gilles Fedak. XtremLab: A System for Characterizing Internet Desktop Grids. In *Poster in The 15th IEEE International Symposium on High Performance Distributed Computing HPDC'06*, Paris, France, June 2006.

[13] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien, and Henri Casanova. On Resource Volatility in Enterprise Desktop Grids. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (eScience'06)*, pages 78–86, Amsterdam, Netherlands, December 2006.

[14] Derrick Kondo, Bahman Javadi, Alex Iosup, and Dick Epema. The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems. In *10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2010.

[15] B. Javadi, D. Kondo, JM. Vincent, and D.P. Anderson. Mining for statistical availability models in large-scale distributed systems: An empirical study of seti@home. In *17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, September 2009.

[16] Oleg Lodygensky, Gilles Fedak, Franck Cappello, Vincent Neri, Miron Livny, and Douglas Thain. XtremWeb & Condor : Sharing Resources Between Internet Connected Condor Pools. In *Proceedings of CCGRID'2003, Third International Workshop on Global and Peer-to-Peer Computing (GP2PC'03)*, pages 382–389, Tokyo, Japan, 2003. IEEE/ACM.

[17] Oleg Lodygensky, Gilles Fedak, Vincent Neri, Cécile Germain, Franck Cappello, and Alain Cordier. Augernome & XtremWeb : Monte Carlo Computation on a Global Computing Platform. In *Proceedings of CHEP03 Conference for Computing in High Energy and Nuclear Physics*, San Diego, USA, 2003.

[18] 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.

[19] Tran Ngoc Minh and Lex Wolters. Towards a profound analysis of bags-of-tasks in parallel systems and their performance impact. In *High-Performance Parallel and Distributed Computing*, 2011.

[20] Alexandru Iosup, Ozan Sonmez, Shanny Anoep, and Dick Epema. The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on High performance distributed computing*, HPDC '08, 2008.

[21] Simon Delamare, Gilles Fedak, Derrick Kondo, and Oleg Lodygensky. SpeQuloS : A QoS Service for Hybrid and Elastic Computing Infrastructures. *Journal of Cluster Computing*, 2013.

[22] Simon Delamare, Gilles Fedak, Derrick Kondo, and Oleg Lodygensky. SpeQuloS: A QoS Service for BoT Applications Using Best Effort Distributed Computing Infrastructures. In *Proceedings of the 21st ACM International Symposium on High Performance Distributed Computing (HPDC'12)*, pages 173–186, Delft, The Netherlands, June 2012.

[23] Simon Delamare and Gilles Fedak. SpeQulos: A Framework for QoS in Unreliable Distributed Computing Infrastructures using Cloud Resources. In *Grid5000 Spring School*, Reims, France, February 2011. **Best Presentation Award**.

[24] Simon Delamare and Gilles Fedak. Towards Hybridized Clouds and Desktop Grid Infrastructures. In Christophe Cérin and Gilles Fedak, editors, *Desktop Grid Computing*, pages 261–285. Chapman & All/CRC Press, 2012.

[25] Oleg Lodygensky, Etienne Urbah, Simon Dadoun, Anthony Simonet, Gilles Fedak, Simon Delamare, Derrick Kondo, Laurent Duflot, and Xavier Garrido. FlyingGrid : from Volunteer Computing to Volunteer Cloud. In *poster in Computing in Hign Energy and Nuclear Physics (CHEP'12)*, New York, USA, 2012.

[26] E. Urbah, P. Kacsuk, Z. Farkas, G. Fedak, G. Kecskemeti, O. Lodygensky, A. Marosi, Z. Balaton, G. Caillat, G. Gombas, A. Kornafeld, J. Kovacs, H. He, and R. Lovas. EDGeS: Bridging EGEE to BOINC and XtremWeb. *Journal of Grid Computing*, 7(3):335–354, September 2009.

[27] Attila Csaba Marosi, Peter Kacsuk, Gilles Fedak, and Oleg Lodygensky. Sandboxing for Desktop Grids Using Virtualization. In *Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing PDP 2010*, pages 559–566, Pisa, Italy, February 2010.

[28] Gabriel Caillat, Oleg Lodygensky, Gilles Fedak, Haiwu He, Zoltan Balaton, Zoltan Farkas, Gabor Gombas, Peter Kacsuk, Robert Lovas, Attila Csaba Maros, Ian Kelley, Ian Taylor, Gabor Terstyanszky, Tamas Kiss, Miguel Cardenas-Montes, Ad Emmen, and Filipe Araujo. EDGeS: The art of bridging EGEE to BOINC and XtremWeb. In *Proceedings of Computing in High Energy and Nuclear Physics (CHEP'09) (Abstract)*, Prague, Czech Republic, March 2009.

[29] Gabriel Caillat, Gilles Fedak, Haiwu He, Oleg Lodygensky, and Etienne Urbah. Towards a Security Model to Bridge Internet Desktop Grids and Service Grids. In *Proceedings of the Euro-Par 2008 Workshops (LNCS), Workshop on Secure, Trusted, Manageable and Controllable Grid Services (SGS'08)*, Las Palmas de Gran Canaria, Spain, August 2008. 247–259.

[30] 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.

[31] David Anderson and Gilles Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pages 73–80, Singapore, May 2006.

[32] Gilles Fedak, Cécile Germain, Vincent Néri, and Franck Cappello. XtremWeb: A Generic Global Computing Platform. In *Proceedings of 1st IEEE International Symposium on Cluster Computing and the Grid CCGRID'2001, Special Session Global Computing on Personal Devices*, pages 582–587, Brisbane, Australia, May 2001. IEEE/ACM, IEEE Press.

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399