QosCosGrid end user tools in PL-Grid - good practices, examples and lesson learned

Mariusz Mamoński [1], Tomasz Piontek[1], Bartosz Bosak[1], Piotr Kopta[1], Krzysztof Kurowski[1]

[1]Poznan Supercomputing and Networking Center, Poland

mamonski@man.poznan.pl, piontek@man.poznan.pl, bbosak@man.poznan.pl, pkopta@man.poznan.pl, krzysztof.kurowski@man.poznan.pl

Abstract. QosCosGrid is a middleware stack being developed at Poznan Supercomputing and Networking Center, which has been initially deployed within the PL-Grid e-infrastructure. After its successful adoption within the Polish research communities, we started collaboration at the European level. The efforts resulted in signing the Memorandum of Understanding with EGI (European Grid Infrastructure) in November 2012 and with BCC (Basic Coordination Centre) of Ukrainian National Grid in August 2013. As far as technological aspects are concerned, the QosCosGrid stack officially entered the UMD (Unified Middleware Distribution) pipeline in September 2013. In this paper we provide a reader with a short introduction to the QosCosGrid architecture and its core components with a strong emphasis on the available end user tools: QCG-Icon - a desktop GUI application that integrates with the operating system and QCG-SimpleClient - a set of command-line tools which breaks the barrier that existing batch systems' users are facing when starting using the Grid. We also share our experience related to end user support and discuss plans for further development works.

Keywords

Grids, EGI, PL-Grid, e-infrastructure, QosCosGrid, Science Gateways, HPC

1 Introduction

The QosCosGrid (QCG)[4] middleware is an integrated system offering advanced job and resource management capabilities to deliver to end users supercomputer-like performance and structure. By connecting many distributed computing resources together, QCG offers highly efficient mapping, execution and monitoring capabilities for variety of applications, such as parameter sweep, workflows, MPI or hybrid MPI-OpenMP. The QosCosGrid middleware provides also a set of unique features, such as advance reservation and co-allocation of distributed computing resources.

In October 2012 a memorandum of understanding was signed between EGI.eu and Poznan Supercomputing and Networking Center[7]. This document was an official step toward sustainable deployment of the QosCosGrid stack into the European grid ecosystem. The collaboration is focused both on integrating the contributed software components into the operational infrastructure, i.e., monitoring, accounting, information, support services and conducting joint dissemination activities.

The document is organized in the following way. Section 2 outlines a generic architecture of QosCosGrid together with a brief description of its main components. In Section 3 we discuss a philosophy standing behind the QosCosGrid and present a few main concepts of the middleware. The next section, Section 4, is devoted to description of two elementary client tools, namely QCG-Icon and QCG-SimpleClient. Section 5 provides a bunch of practical information regarding the cooperation with end users. General description of the outcomes, lessons learned as well as plans for future development are raised in Section 6. Finally, in Section 7, we summarize discussed topics and draw the conclusions.

2 QosCosGrid Architecture

The QosCosGrid middleware consists of two logical layers: Grid and local one. Grid-level services control and supervise the whole process of execution of experiments which are spread between independent administrative domains. The administrative domain represents a single resource provider (e.g. datacenter) participating in a certain Grid environment by sharing its computational resources. The main component of the grid layer is the QCG-Broker metascheduling service whereas the QCG-Computing and QCG-Notification acts on a local level. We present the overall architecture of the system in Figure 1.

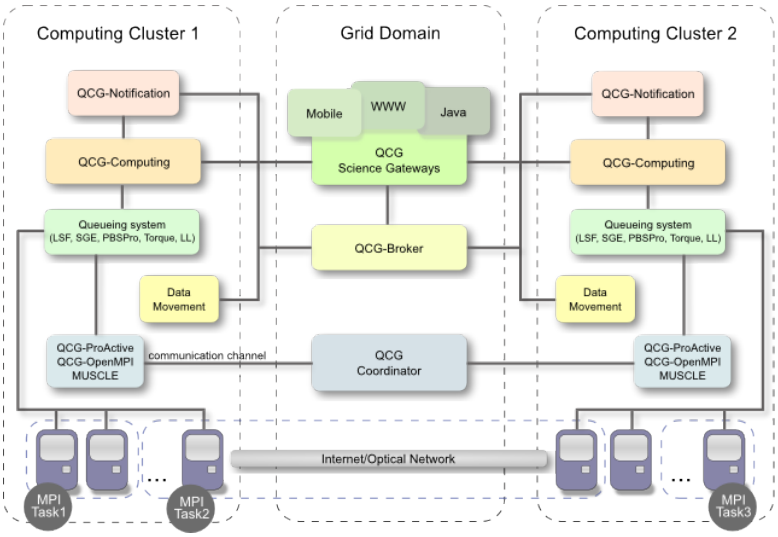

Figure 1. QosCosGrid overall architecture.

Below we shortly summarize functionality of the three major QosCosGrid services:

- QCG-Computing - The core local-level component, usually deployed on access nodes of the batch systems (like Torque or SLURM) provides remote access to task submission and advance reservation capabilities of local batch systems via interface compatible with the OGF HPC Basic Profile**??** specification. The integration with the queuing system was realized using the DRMAA[10] interface. Moreover it offers basic file transfer mechanisms utilized by QCG-Icon and a built-in information service that provides comprehensive dynamic information about the current cluster status to the QCG-Broker service.
- QCG-Notification - The main asynchronous message bus between the services, applications and the end users. It supports the topic-based publish/subscribe pattern for message exchange. It is capable of sending notifications using variety of transport mechanism, including HTTP, SMTP and, what is a unique feature, the XMPP protocol.
- QCG-Broker - The grid-level component of the QCG stack that controls, schedules and generally supervises the execution of tasks including preparation of the execution environment and transferring of results. QCG-Broker is based on dynamic resource selection, mapping and advanced scheduling methodology, combined with feedback control architecture. It deals with dynamic Grid environment and resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support. Its capabilities are exposed to the end user by the means of the QCG-SimpleClient tools.

3 Common Concepts

In this chapter we discuss a few other important aspects of the QosCosGrid design and its operational philosophy that relates to the applications encapsulation, monitoring and resources selection.

3.1 Applications Wrappers

Making application submission an opaque process, no matter where it runs, was one of the foundations of the Grid Computing. However meeting this requirement requires from the Grid Middleware dealing with unavoidable heterogeneity of resources composing the grid system. The same application can be installed in various locations on different systems. Moreover the scratch file system locations can also differ among the systems and the way how the application is spawned may not be the same. Some of these problems can be solved with the help of Environment Modules [5], however still the module/environment variables names may be not coherent among the sites. For this reason QosCosGrid introduces abstract notion of an application. With this approach an application name (e.g. GROMACS) is mapped locally to the full path of the application's wrapper script. The wrapper script handles application execution, i.e. it loads proper module, if needed changes to the scratch directory, spawns application and remove temporary files after it terminates. For some of applications, like Gaussian, the input file is automatically preprocessed so the number of threads and max available memory are set accordingly with resources that were allocated to the job. What is worth mentioning, in QosCosGrid, we separated the script logic (which is global and updated periodically) from the script configuration (which is local).

3.2 Application Level Monitoring

During course of years we have learned how valuable it is to the user to get detailed status of her/his simulations. Having access to the produced data only at the end of job is acceptable only for a very short runs. For this reason QCG give a user possibility to peek output of any of his running jobs. Moreover it is possible to establish interactive session (using qcg-connect command) with an already started batch job. After the interactive session has been established a user can list job directory and its subdirectories, view any file or run ps/top commands to see if program is not hanging or swapping memory. Moreover many of these erroneous situations can be detected by just observing dynamic job metrics displayed in the QCG tools, namely CPU efficiency and memory usage. Other commonly asked by end users question is "What time my job will start?". The QosCosGrid services try to answer this question by extracting this information from the local scheduler. Although this is a best effort metric, it gives user at least a rough estimation of expected waiting time. Finally the QCG job environment allows registering notifications on application's output, i.e. whenever given phrase appears in the output file (for e.g. ENERGY) the system sends notification, using either e-mail or XMPP protocol, directly to the end user. Moreover it is possible to monitor the progress of application in dedicated web portal, which presents data in a graphical way base on the predefined template. The Figure 2 presents the visualization of energy changes in an example Gaussian simulation.

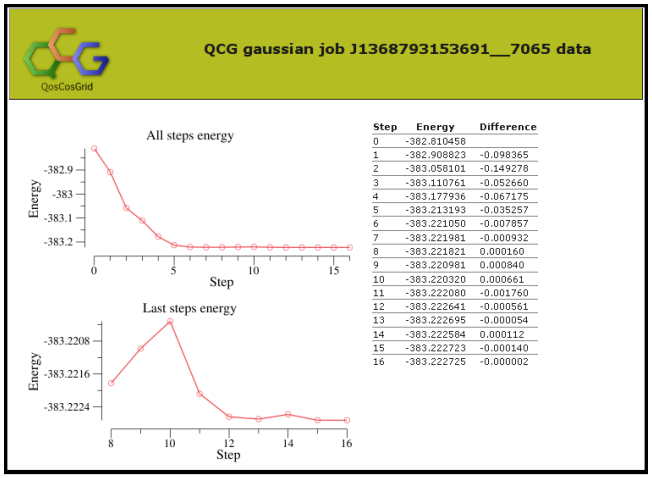

Figure 2. Graph showing progress of an example gaussian simulation.

3.3 Brokering vs Resource Discovery

When submitting jobs to the grid environment users expect that they will be started and will provide results as quickly as possible. In the QosCosGrid stack realization of this need is part of the functionality of the

specialized service called QCG-Broker. The service assigns jobs to clusters in a way that minimizes the time which jobs stay in queues waiting for resources. The decision to which cluster submit a job is taken based on the current status of the whole system returned by all currently active instances of QCG-Computing services. The brokering algorithm implemented in QCG-Broker includes two logical steps. In the first step clusters that do not meet user or system requirements are excluded from the list of potential sites. The cluster to be accepted must to pass all verification criteria including, among others, accessibility for the given user and grant, presence of sufficient number of nodes of requested characteristic, presence of requested applications and software modules or support for advance reservation. The clusters that passed the first step verification are graded to find the optimal one. The evaluation is performed based on the weighted sum of a set of metrics calculated for every cluster. The expendable list of plug-ins with configurable weighs allows to tailor the brokering policy to the specific system and assigns task to resources in the way that satisfies Users (Job Owners) and their applications requirements as well as constraints and policies imposed by other stakeholders, i.e. resource owners and Grid administrators.

The Table 1 presents the subset of possible grading plug-ins with their defaults weights.

Table 1. QCG-Broker scheduler plugins.

Plugin name	Default Weight	Description
RandomGrading	2	grades clusters in a random manner
SlotGrading	10	grades cluster based on free slots/total slots ratio
FreeNodeGrading	10	prefers clusters with more completly free nodes
NodesNumberGrading	1	prefers clusters with higher number of nodes
QueuesGrading	5	takes into consideration ratio between running and pending jobs
WaitingTimeGrading	5	grades cluster based on average waiting time of all already started jobs present in the system
LRUGrading	3	Last Recently Used - prevents submitting all jobs to a single cluster

As an alternative we also provided users with the qcg-offer tool. It is a command-line tool that allows regular users to query about free resources available in the grid. It leverages the same, fine-grained, information provided by the QCG-Computing services as QCG-Broker. It is possible to query single site, show full or aggregated view of cluster nodes or filter resources based on available memory, total/free number of cores, nodes attributes, etc. User can later utilize this information and his own experience to select target resource adjusting job's size and topology by changing number of requested nodes and/or slots per node. Moreover the QCG-Offer tool is capable of searching for applications and modules installed on all sites. An example working session with the QCG-Offer command is shown in Figure 3.

```
[plgmamonski@qcg ~]$ qcg-offer site=hydra
HYDRA:
Summary:
    Metric Name        nodes/cores      share
     Total Resources:    282/5340    100%/100%
        Up Resources:    225/4340     79%/81%
      Used Resources:    119/1965     42%/36%
      Free Resources:     99/1396     35%/26% (FreeNodes=87x12,7x16,5x48)
  PartFree Resources:    129/2231     45%/41% (AvgFreeCoresPerNode=17)
  Reserved Resources:      2/24        0%/00% (Utilization=0%)
```

Figure 3. An example session with the qcg-offer command.

4 End User Tools

In this chapter we will describe the two most popular end users tools used by the QosCosGrid users: QCG-SimpleClient - a set of command-line tools which breaks the barrier that existing batch systems' users are facing when migrating to the Grid and QCG-Icon - a desktop GUI application that integrates with the operating system.

4.1 QCG-SimpleClient

The QCG-SimpleClient is a set of command line tools, inspired by the simplicity of batch system commands. The commands allow user to submit, control and monitor large number of grid batch jobs as well as to reserve resources to obtain requested quality of service. The learning effort needed to start using QCG-SimpleClient is relatively small as the commands are modeled after the ones known from queuing systems. Additionally the job description file is a plain BASH script annotated with #QCG directives what is also common approach for all nowadays queuing systems. The main difference is that user has to specify explicitly all files and directories that have to be staged in/out as there is no global shared file system between all sites. Fortunately the staging directives accepts also short relative local paths beside the full URLs. Listing 1 presents an example of the QCG-SimpleClient job description.

```
#QCG note=NAMD apoa1
#QCG host=hydra.icm.edu.pl
#QCG walltime=PT10M
#QCG queue=plgrid
#QCG nodes=1:12:12
#QCG output=apoa1.output
#QCG error=apoa1.error
#QCG application=NAMD
#QCG argument=apoa1/apoa1.namd
#QCG stage-in-file=apoa1.zip
#QCG preprocess=unzip apoa1.zip
#QCG stage-out-dir=. -> results
#QCG notify=xmpp:tomasz.piontek@plgrid.pl
#QCG watch-output=mailto:tp@mail,20,ENERGY
```

Listing 1: An example QCG-SimpleClient submission script.

One of the most frequently requested functionality that has been recently added to the QCG-SimpleClient is support for interactive tasks. The user can either run any commandline application in an interactive mode or to get interactive access to system not offering such functionality on queuing system level. The functionality is often used by users to compile their own code or to run some test/debugging sessions on systems offering access only via middleware services.

4.2 QCG-Icon

QCG-Icon is a desktop application written specifically for the Windows platform, but available also for Linux and Mac OSX distributions. It was designed to enable the selected applications installed on the computing resources of the PL-Grid infrastructure, and available through the QosCosGrid services. While developing QCG-Icon the special emphasis was put on the following fact: using an application installed in the grid environment should be as intuitive as using a locally installed application. At the moment, QCG-Icon supports large portfolio of applications, including MATLAB, R, NAMD, Gaussian (integrated also with GaussView), GAMESS, Molpro, LAMMPS, Quantum ESPRESSO, Crystal09, NWChem, GROMACS and CPMD. Any other application can be also run as soon as a proper BASH script is provided. Despite its simplicity QCG-Icon exposes most of the functionality offered by the QosCosGrid stack, including parallel jobs, live output monitoring and providing online statistics about the job's resources usage. The overview of the QCG-Icon graphical user interface is shown in Figure 4.

Quite recently a LCAS/LCMAPS[2] based authorization plugin has been developed for the QCG-Computing service. This work enabled QosCosGrid middleware to support authorization mechanism based on Virtual Organization Management Support (VOMS) infrastructure [1]. Moreover the QCG-Icon application had to be extended to generate a VOMS proxy certificates when configured for a non PL-Grid virtual organization. The newly developed capability facilitates adoption of the QosCosGrid stack by existing Virtual Organizations and resource providers. The first external Virtual Organization which was integrated with the QosCosGrid services on selected resources was Gaussian VO[1].

[1] https://voms.cyf-kr.edu.pl:8443/voms/gaussian

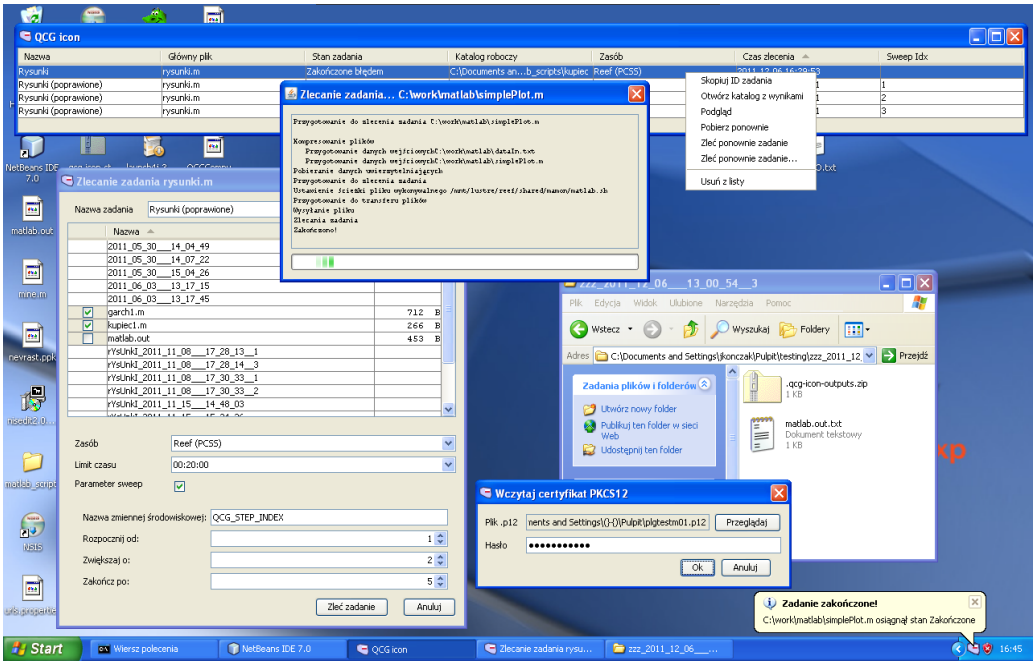

Figure 4. QCG-Icon graphical user interface.

5 End Users Support and Feedback

The standard support for QosCosGrid stack and end users tools in the PL-Grid e-infrastructure is offered via the official project's ticket tracking system[2]. A separate queue has been created in the system on the sole purpose of handling QCG related issues. Moreover we offer users a less formal communication channel realized by a separate e-mail address qcg@plgrid.pl, which is an alias for every member of the QCG team. This is a common way to request new functionality or to ask general questions about the QosCosGrid stack. What is also important we try to be proactive and we collect on daily basis and in a semi-automatic manner information from all QCG services. We further search the gathered data for an erroneous situations and, if needed, contact the user offering her/him additional help.

6 Lesson Learned and Further Work

After almost two years of working with production e-infrastructures we have learned that maintaining services, applications and supporting end users can be as time consuming as code development itself. The infrastructure evolves all the time and due to its large scale single site failures are nothing uncommon. For these reasons in the next months we plan to put more efforts on developing functionalities that make QosCosGrid even more sustainable, maintainable and user-friendly. In particular we plan to:

- Provide administrators with the automatization of QCG installation and configuration by the means of the Puppet configuration management system[8].
- Implement automatic mechanism of detection applications' versions installed on the cluster as currently all QCG-enabled applications must be configured manually. This functionality would relay on environment modules [5] - currently the most widely adopted solution for management of installed applications in the HPC systems.
- Develop a new, more comprehensive set of Nagios[6] probes, that would automatically test not only a simple batch job but every QCG-enabled application offered on the given resource. This will provide automatic regression testing of the constantly changing infrastructure.
- Implement additional communication channel from the local site admins and the QCG team to the end users that would be used to inform about the hardware or system failures and system modifications

[2]https://helpdesk.plgrid.pl/

- Finish integration with the PL-Grid X.509 certificates and keys distribution system - KeyFS[9] so an end user would experience only the password based authentication which is more widespread outside the grid world.
- Improve error messages reported by QosCosGrid so they would provide information for the user what action should be taken to mitigate the problem, potentially pointing to the user's manual.
- Run periodical surveys on QCG end users tools.

We believe that all the work will pay back in reduced maintenance costs and better user experience.

7 Conclusions

The QosCosGrid is used on daily basis by many researchers in Poland coming from various research domains such as quantum chemistry, nanotechnology, metallurgy, astrophysics and bioinformatics. It is currently the first middleware in PL-Grid counting the cpu hours consumed by its users (Figure 5). This success would not be possible without offering simple but powerful end users tools and comprehensive end users support.

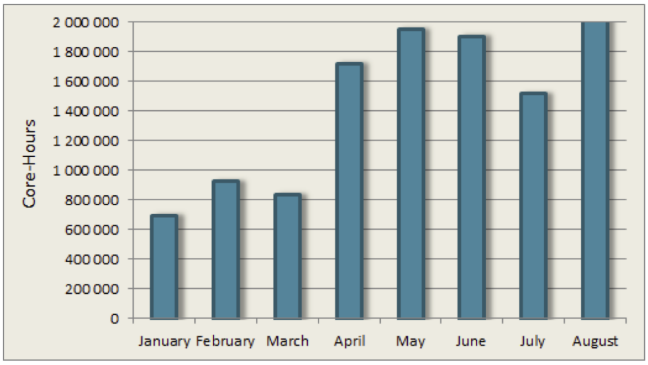

Figure 5. Total CPU-hours consumed by the QCG users on PL-Grid Infrastructure in 2013.

We are currently finalizing most of the milestones defined by the Memorandum of Understanding between EGI and PSNC. The QosCosGrid services have been already integrated with the EGI monitoring, accounting, helpdesk and dashboard systems. The implementation of information systems compatible with BDII[3] is under way. Moreover, in September 2013, all the core components of the QosCosGrid stack became part of UMD (Unified Middleware Distribution).

Last, but not least, in August 2013 a Memorandum Of Understanding was signed between Poznan Supercomputing and Networking Center and BCC (Basic Coordination Centre) of Ukrainian National Grid[3]. One of the major objective of the MoU is to "provide robust, well-designed, user-centric services to scientific user" based on the QosCosGrid services. That will be the first QosCosGrid deployment in such scale outside Poland.

8 Acknowledgments

This research was supported in part by PL-Grid Infrastructure. We also wish to thanks all our PL-Grid colleagues that contributed into QosCosGrid deployment, testing and dissemination activities.

References

[1] Roberto Alfieri, Roberto Cecchini, Vincenzo Ciaschini, Luca dell'Agnello, Akos Frohner, Alberto Gianoli, Karoly Lorentey, and Fabio Spataro. Voms, an authorization system for virtual organizations. In Grid computing, pages 33–40. Springer, 2004.

[3] twiki.cern.ch/twiki//bin/view/EGEE/BDII

[2] Roberto Alfieri, Roberto Cecchini, Vincenzo Ciaschini, A Gianoli, F Spataro, F Bonnassieux, P Broadfoot, G Lowe, L Cornwall, J Jensen, et al. Managing dynamic user communities in a grid of autonomous resources. arXiv preprint cs/0306004, 2003.

[3] BCC. The basic coordination centre of ukrainian national grid, 2013.

[4] Bartosz Bosak, Jacek Komasa, Piotr Kopta, Krzysztof Kurowski, Mariusz Mamoński, and Tomasz Piontek. Building a national distributed e-infrastructure - pl-grid. chapter New capabilities in qoscosgrid middleware for advanced job management, advance reservation and co-allocation of computing resources — quantum chemistry application use case, pages 40–55. Springer-Verlag, Berlin, Heidelberg, 2012.

[5] John L Furlani. Modules: Providing a flexible user environment. In Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V), pages 141–152, 1991.

[6] Emir Imamagic and Dobrisa Dobrenic. Grid infrastructure monitoring system based on nagios. In Proceedings of the 2007 workshop on Grid monitoring, pages 23–28. ACM, 2007.

[7] European Grid Initiative. New middleware for new communities, 2011.

[8] Luke Kanies. Puppet: Next-generation configuration management. The USENIX Magazine. v31 i1, pages 19–25, 2006.

[9] Marcin Teodorczyk and Bartłomiej Balcerek. Making x.509 certificates simple to use in pl-grid project, 2011.

[10] Peter Troger, Hrabri Rajic, Andreas Haas, and Piotr Domagalski. Standardization of an api for distributed resource management systems. In Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on, pages 619–626. IEEE, 2007.