New QosCosGrid Middleware Capabilities and Its Integration with European e-Infrastructure

Bartosz Bosak, Piotr Kopta, Krzysztof Kurowski, Tomasz Piontek, Mariusz Mamoński†

Poznan Supercomputing and Networking Center
{bbosak,pkopta,krzysztof.kurowski,piontek}@man.poznan.pl

Abstract. QosCosGrid (QCG) is an integrated system offering leading job and resource management capabilities in order to deliver supercomputer-like performance and structure to end users. By combining many distributed computing resources together, QCG offers highly efficient mapping, execution and monitoring capabilities for a variety of applications, such as parameter sweep, workflows, multi-scale, MPI or hybrid MPI-OpenMP. The QosCosGrid middleware also provides a set of unique features, such as advance reservation, co-allocation of distributed computing resources, support for interactive tasks and monitoring of a progress of running applications. The middleware is offered to end users by well-designed and easy-to-use client tools. At the time of writing, QosCosGrid is the most popular middleware within the PL-Grid infrastructure. After its successful adoption within the Polish research communities, it has been integrated with the EGI infrastructure and through a release in UMD and EGI-AppDB it is also available at European level. In this chapter we focus on extensions that were introduced to QosCosGrid during the period of the PL-Grid and PLGrid PLUS projects in order to support advanced user scenarios and to integrate the stack with the Polish and European e-Infrastructures.

Keywords: grid computing, middleware, advance reservations, co-allocation of resources, application monitoring, notifications

1 Introduction

In the last years, there have been identified two distinctive groups of end users interested in obtaining efficient access to computational resources belonging to the national- or European-level e-Infrastructures such as PL-Grid and EGI. Both groups of users differ primarily in their experience in running jobs on HPC clusters and thus they expect different kinds of tools tailored to their needs, habits and aforementioned levels of experience. The first group consists of researchers

usually having computer science background and familiar with cluster solutions. They expect command-line tools similar to these known from queuing systems, but offering access to all resources of e-Infrastructure. The second group consists of domain-oriented researchers, who have not yet used clusters but who are willing to migrate from their desktop systems and to take full advantage of HPC computing. These users want to solve bigger instances of problems or to parallelize the execution of numerous application runs. They require intuitive tools resembling the tools they are accustomed to and braking the technology barrier associated with the migration to the grid environment, in order to exploit them in their daily work.

To support the two groups of users the QosCosGrid (QCG) middleware, developed in Poznan Supercomputing and Networking Center (PSNC), delivers an advanced multi-layered e-Infrastructure which successfully integrates primary and brand-new services and tools capable of dealing with various kinds of computationally intensive simulations. Recently, all these services and tools were integrated with European e-Infrastructure, and were made available through the EGI distribution channels, like Unified Middleware Distribution (UMD) [32] and EGI Applications Database (AppDB) [25] for the research communities outside PL-Grid.

Within this document we present new capabilities of the QosCosGrid middleware developed in the time frame of the PLGrid PLUS project that extend the core functionality described in [8]. The document focuses on extensions provided to the basic QCG services as well as it describes several recently developed components that support new user scenarios and integrate QCG solutions with the Polish and European e-Infrastructures.

The remaining part of the chapter is organized as follows: Section 2, Related Work, provides a brief overview of existing grid middleware and available end user tools. The main objectives underlaying the QosCosGrid middleware are presented in Section 3. Section 4 presents a high-level architecture of QosCosGrid along with a short description of its main components. In the next section we outline new features of the middleware, i.e. application scripts, improved brokering, advance reservation, co-allocation of resources and support for notifications. Section 6 describes the main QosCosGrid client tools, namely QCG-SimpleClient and QCG-Icon, and introduces concepts of QCG-Data and QosCosGrid Science Gateway. The next section reports the status of the integration process of QosCosGrid middleware with the EGI infrastructure. Finally, a general discussion on the outcomes achieved so far and the plans for future development of the middleware are addressed in Section 8.

Acknowledgement

This publication was published as a part of the eScience on Distributed Computing Infrastructure book. The final publication is available at `http://link.springer.com/chapter/10.1007%2F978-3-319-10894-0_3`

2 Related Work

The QosCosGrid is one of several advanced middleware distributions deployed on e-Infrastructures. Although each middleware provides an exhaustive set of functions required to efficiently run advanced simulations, the decision to select a particular system for grid calculation is complex as it usually entails the way how the scientific work will look like during a long period of time. Middlewares differ in complexity, applied technologies, offered functions as well as in character of end user tools. In some cases, small differences may cause serious problems or bring significant benefits in the future.

Undoubtedly, each of middlewares has its loyal users, who historically or/and politically started to built their application scenarios based on specific environments and have no reason to change the selected system. A clear example may be the gLite framework massively exploited by the community associated with CERN LHC for their simulations. [17] Other groups of users, connected by specific projects, requirements or geographical location decided upon different middleware technologies, e.g. UNICORE [4][5], ARC [16] or SAGA [31]. It is worth noting that the support for advance reservations and co-allocation of resources is not a common functionality and is available only in few middlewares by adoption of specialized frameworks, such as HARC [18] or GridARS [22].

In turn, a number of various client programs that allow running computations using one or many middleware systems is relatively large. Usually, a middleware introduces its own command-line tool(s), e.g. UNICORE provides UNICORE CommandLine Client [21], gLite provides its Command Line Interface [17], but often it also offers dedicated GUI applications, like UNICORE RichClient [11]. There are also advanced programs, capable of submitting jobs to many middlewares, such as Migrating Desktop and gEclipse [19]. A different group of high-client tools includes web solutions, e.g. GridSpace2 [10] or science-gateways [12].

To help users select the optimal middleware, the work [8] provides basic comparison of the QosCosGrid middleware with gLite and UNICORE.[1]

3 Goals of QosCosGrid

The QosCosGrid (QCG) middleware could be seen as a system that hides the complexity of many heterogeneous computing resources behind a single, intuitive and user-friendly interface. However, the functionalities offered by QosCosGrid are not simply limited to a direct mapping of particular functions available in queuing systems into the well-designed interface at a grid level. The aim was to design and implement a system driven by real requirements and expectations of researchers. Therefore, from the beginning, the development of QCG services and tools is carried out in a close collaboration with research groups representing various domains of science. This resulted in developing a system adjusted to

[1] The comparison presents state for year 2012

specific needs and habits of scientific users, and provisioning functions often unavailable in queuing systems and other grid middlewares. The list presented below outlines only some of them:

- "intelligent" brokering capabilities,
- support for advance reservations,
- co-allocation of heterogeneous computing resources,
- cross-cluster execution of jobs,
- support for multiscale computing,
- support for interactive tasks,
- flexible monitoring capabilities,
- intuitive command-line tools,
- user-friendly graphical interfaces.

The particular features of QosCosGrid are described in details in next sections.

4 High-level QosCosGrid Architecture

Basically, 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 among independent administrative domains. One administrative domain represents a single resource provider (e.g. data center) participating in a certain grid environment by sharing its computational resources. The main component of a grid layer is the QCG-Broker metascheduling service whereas QCG-Computing services play the main role at a local level. We present the overall architecture of QosCosGrid system in Figure 1.

QosCosGrid components, with respect to their function and placement in the architecture, may be divided into several groups marked on the left side of the diagram. On that basis, the description presented below aims to summarize the role of particular items building the whole QosCosGrid environment.

Infrastructure
QosCosGrid realizes an access to computing resources using LRMS/batch systems (e.g. Torque or SLURM). The integration between QCG services, i.e. QCG-Computing, and the underlying batch system is provided with the use of the DRMAA interface [23].

Applications and tools
The QosCosGrid middleware is able to run practically every application installed on resources, including cross-cluster applications based on MUSCLE, MPI or ProActive libraries. In order to simplify the way of running popular applications, the middleware provides a number of scripts and tools that wrap certain commands.

Cluster-level services
At a cluster level, or more generally - local administrative level, QCG is represented by QCG-Computing, QCG-Accounting and QCG-Notification usually deployed together on access nodes of batch systems.

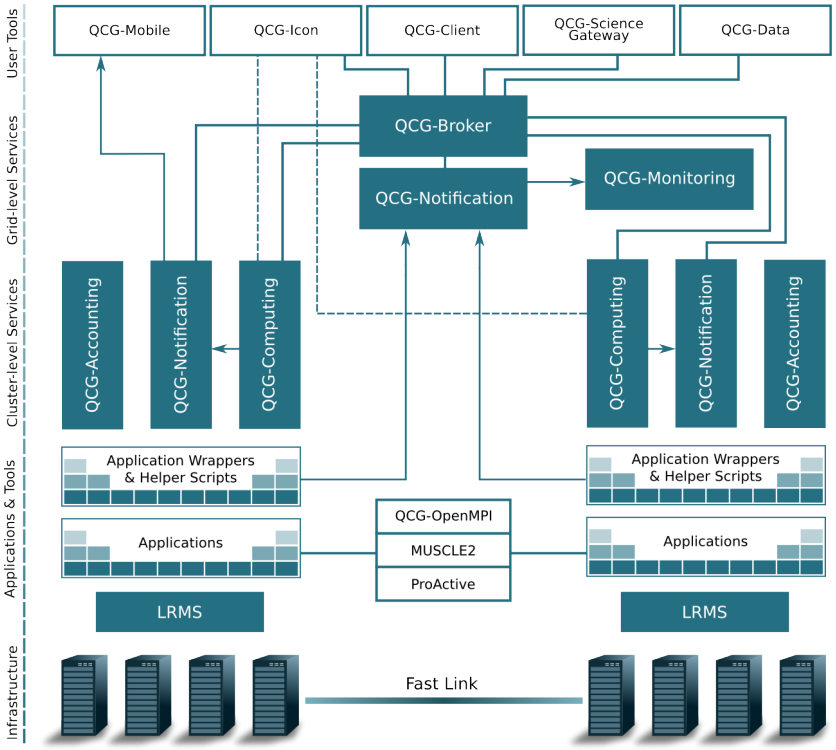

Fig. 1. The high-level architecture of QosCosGrid

QCG-Computing is a core service that provides a remote access to task submission and advance reservation capabilities of local batch systems via interface compatible with the OGF HPC Basic Profile specification [27]. As mentioned earlier, the QCG-Computing service is integrated with the underlying queuing system using the DRMAA interface. QCG-Computing offers basic file transfer mechanisms utilized by QCG-Icon and includes built-in information service that provides QCG-Broker with comprehensive dynamic information about the current cluster status.
QCG-Accounting is a tool that publishes usage records to the external accounting systems. Until now, it has been integrated with the three accounting systems, i.e. EGI accounting system called APEL [15], the PL-Grid one called BAT [8] and the Grid-SAFE [26].
QCG-Notification plays the role of the main asynchronous message bus between the services, applications and end users. It is deployed at both local and grid level. The service supports the topic-based publish/subscribe pattern for message exchange defined by the Oasis WS-Notification standard [30]. QCG-Notification is capable of sending notifications using a variety of transport mechanisms, including HTTP/HTTPS, SMTP (e-mail) and XMPP protocol.

Grid-level services

The group of grid level components, in addition to QCG-Notification, includes two additional services, namely QCG-Broker and QCG-Monitoring.
QCG-Broker controls, schedules and generally supervises the execution of tasks, including preparation of the execution environment and transferring the results. This key service is based on dynamic resource selection, mapping and advanced scheduling methodology, combined with the feedback control architecture. It operates within a dynamic grid environment and deals with resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support.
QCG-Monitoring is a new grid-level service built on top of the QCG-Notification system. It offers end users a possibility of monitoring a progress of the application execution in a dedicated web portal. After processing, the application progress is displayed in a graphical way as a set of tables and charts in accordance with the selected predefined template.

User tools

Capabilities of the QosCosGrid middleware are offered to end users by means of a number of client-tools, including command-line interface: QCG-SimpleClient, desktop GUI programs: QCG-Icon, and QCG-Data, mobile application: QCG-Mobile, as well as high-level web-based solutions like GridSpace2 [6][10] and QCG-ScienceGateway [12].

The detailed description of the core QosCosGrid services can be found in [8].

5 Primary QosCosGrid Functionalities

The objectives of the PL-Grid and PL-Grid PLUS projects have been defined in a way to significantly improve the collaboration between the QosCosGrid developers, end users and computing resource representatives. Owing to the given opportunities, it was feasible to organize frequent talks and discussions in order to improve the middleware, to prepare requested extensions and to implement better client tools. Within this section we present several functionalities of QCG developed primarily under the umbrella of the PL-Grid/PLGrid PLUS projects.

5.1 Application Scripts

Making application submission a transparent process and hiding its details from the user, regardless of where the application actually runs, was one of the foundations of grid computing. However, meeting this requirement implies operating within unavoidable heterogenity of resources composing the grid system by the grid middleware. 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 [13], however, the module/environment variables' names may not be

coherent among the sites. For this reason, QosCosGrid introduces an extra layer of computational resources with the abstract notion of an application. Within this approach an application name (e.g. GROMACS) is mapped locally to the full path of an application's wrapper script. The wrapper script handles application execution, i.e. it loads a proper module, if needed, changes to scratch directory, spawns application and removes temporary files after the application terminates. For some applications, like Gaussian, the input file is automatically preprocessed so that the number of application's threads and maximum 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).

5.2 Improved Brokering Capabilities

When submitting jobs to the grid environment, users expect that their applications will be started on a proper class of worker nodes and will provide results as quickly as possible. In the case of the QosCosGrid stack, the realization of this need is a part of a functionality of a specialized service called QCG-Broker. The service assigns jobs to clusters in a way that minimizes the time in which jobs stay in queues waiting for resources. The decision to which cluster should be submitted a job is made based on a 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. To be accepted, a cluster has to meet all verification criteria including, among others, accessibility for a given user and grant, presence of a 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 of the verification are graded. This evaluation is performed on the basis of the weighted sum of a set of metrics calculated for every cluster. An extensive list of plug-ins with configurable weights allows an infrastructure administrator to tailor the brokering policy to the specific system and to assign tasks to resources in the way that satisfies users (job owners) and meets their application requirements as well as takes into consideration constraints and policies imposed by other stakeholders, i.e. resource owners and grid administrators.

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

5.3 Advance Reservation and Co-allocation of Resources

The QCG middleware, to the best of our knowledge, as the first one, has offered advance reservation capabilities in a production environment. Advance reservation mechanism is exploited to provide end users with the following functionalities: reservation of resources to guarantee requested quality of service and co-allocation of distributed heterogeneous resources to synchronize execution of

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

cross-cluster applications. QCG can automatically search, within a user-defined time window, for free resources for a requested period of time. Within QCG, it is possible either to reserve a given number of slots on any number of nodes or to request for a particular topology by specifying a number of nodes and slots per node. At present, advance reservations can be created and managed using either command-line tools (the QCG-SimpleClient client) or graphical, calendar like, web application (the reservation portal called QCG-QoS-Access) presented in Figure 2. Currently, in QCG, advance reservations are created by calling the LRMS scheduler commands directly, while in the future a leverage of Advance Reservation API of Open Grid Forum DRMAA 2.0 specification [24] is planned. An extensive summary characterizing the concept of advance reservations can be found in Chapter [CROSS-REF to the article about reservations]

The QosCosGrid's support for advance-reservation and co-allocation of various types of resources provides a good opportunity to create complex scenarios consisting of many demanding application modules. Within the MAPPER project [3], the QosCosGrid stack has been integrated with Multiscale Coupling Library and Environment (MUSCLE) [7] which enables cross-cluster execution of so-called multiscale applications. The common multiscale application consists of a number of single-scale modules that calculate some phenomena at different spatial or temporal scales and simultaneously exchange information with one another. Since the elementary modules can be written in different languages and have different resource requirements, the QosCosGrid ability to combine many clusters into the single virtual machine is crucial.

5.4 Application Status and Progress Notifications

Time needed to perform a simulation can differ in the cases of various input parameters and data, but even for the same ones, it can be unpredictable in complex

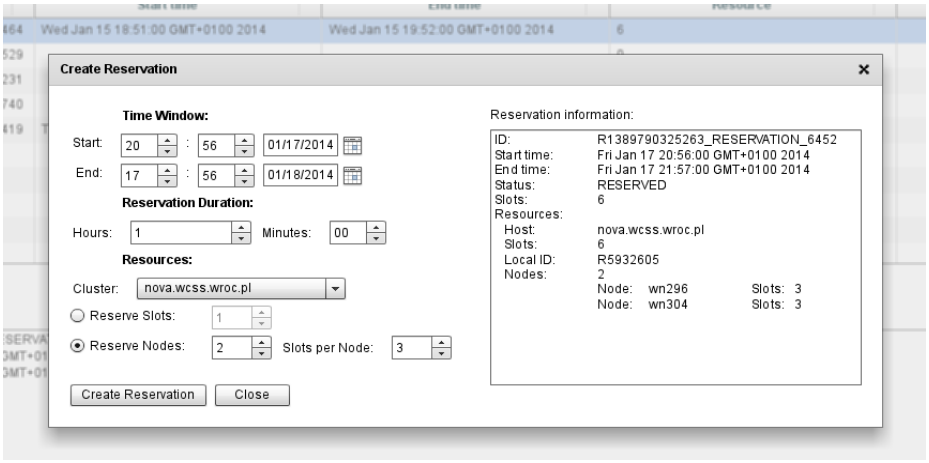

Fig. 2. The QCG-QoS-Access - Reservation Portal

and heterogeneous environments. Within the PL-Grid, for example, the waiting time needed to start a job can be dependend on the current load, while the execution time may be associated with the worker-node and processor type. This non-deterministic relation might be an obstacle for end users who often need to know in advance when they can expect results or which percent of the simulation has already been performed. Moreover, especially for long-running simulations, it is important to know if the execution is performed properly and if produced partial results are correct in order to avoid aimless consumption of resources. Taking into account the above needs, the QosCosGrid middleware provides special notification capabilities. With the help of the QCG-Notification service and its support for e-mail and XMPP protocol, as well as QCG-Monitoring functionality, changes in application execution may be immediately reported to interested parties.

Users are provided with two basic types of notifications, respectively:

1. Notifications of a job status — users may register for obtaining e-mails or XMPP messages informing about a current status of their jobs (e.g. PENDING, RUNNING, FINISHED); whenever the job changes its state, the corresponding notification is generated and sent.
2. Notifications with an application's excerpt — users can also be provided with monitoring data consisting of a certain application's output, i.e. when a given phrase appears in the output file (e.g. "ENERGY=500") of the application, the system generates appropriate notification. The application's excerpt notifications may be sent directly to users via e-mail or XMPP protocol, or alternatively, forwarded to the QCG-Monitoring service that is described later.

The procedure of the registration on notifications is simple and is performed with the use of the QCG command line client. The syntax of QCG-Simple pro-

vides several intuitive directives which, if used, impose flow of certain types of notifications to the specified recipient.

Dedicated Monitoring Solutions
To address specific needs of users, the QCG-Monitoring service was designed and deployed on the top of the QCG notification system. The service offers end users a possibility of monitoring the progress of an application execution in a dedicated web portal. The application progress is displayed in a graphical way in a form of a set of tables and charts in accordance to the predefined template. Users can select from a set of general-purpose templates, they can also utilize templates for quantum chemistry and astrophysics applications that have been prepared in cooperation with domain-oriented researchers. The Figure 3 presents the visualization of energy changes in an example Gaussian simulation.

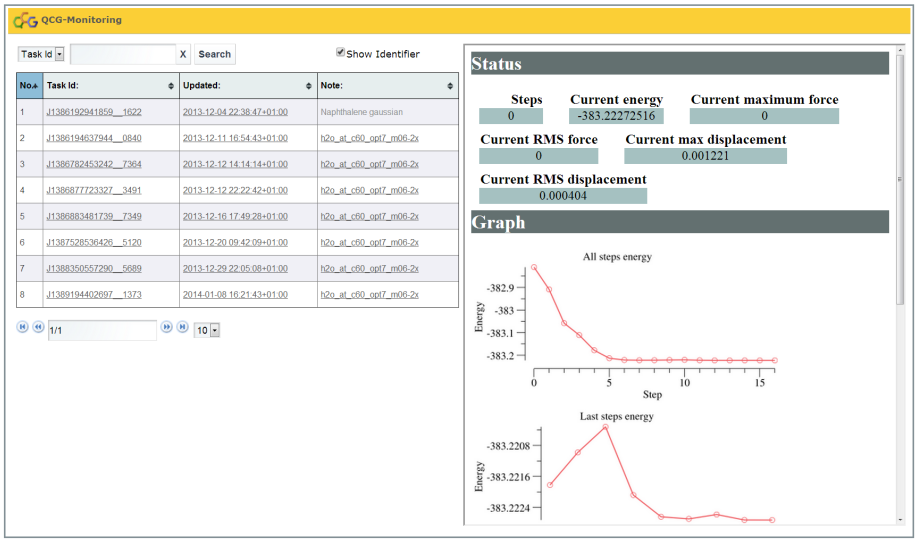

Fig. 3. The QCG-Monitoring portal

Mobile phone and tablet users may benefit from other QCG application called QCG-Mobile which is available for the Android system. This application makes use of XMPP notifications and may be helpful in simple tracking jobs, especially when users do not have an access to theirs PC's.

6 End User Access Tools

In this section we describe the most popular client programs used by QosCosGrid users within the PL-Grid infrastructure, i.e. QCG-SimpleClient - a set of

command-line tools, largely appreciated by the group of existing batch's systems' users, and QCG-Icon - a desktop GUI application suitable for users who have no particular knowledge about clusters or require a handy tool for accessing large computing resources. This section also presents our recent implementation, namely QCG-Data, as well as it shortly characterizes the concept of QCG-ScienceGateway.

6.1 QCG-Simple Client

The QCG-SimpleClient is a tool recommended for all users who do not require advanced capabilities of the QCG middleware like workflows, parallel jobs with topologies or parameter sweep jobs which functionality is usually supported by domain-oriented, dedicated web-based QCG-ScienceGateways. In return, QCG client offers an access to the most frequently used functionalities in a very simple and intuitive way. The QCG-SimpleClient is a set of command line tools, inspired by the simplicity of batch system commands. The tools are dedicated for end users familiar with queuing systems and preferring command line prompt over graphical interfaces. Learning effort needed to start using QCG-SimpleClient is relatively small as commands are modeled in a way similar to the ones known to users from batch systems. The commands allow user to submit, control and monitor a large number of various types of grid batch jobs as well as to reserve resources to obtain requested quality of service. The full list of qcg-* commands is presented below with separation into three groups related to tasks, reservations and state of the system, respectively.

Submission and control of tasks:

- `qcg-cancel` - cancel task(s),
- `qcg-clean` - clean the working directories of given tasks,
- `qcg-connect` - connect with an interactive session to the task,
- `qcg-info` - display detailed information about the given tasks,
- `qcg-list` - list tasks in the system,
- `qcg-peek` - display ending of (stdout, stderr) streams,
- `qcg-proxy` - create a user proxy certificate,
- `qcg-refetch` - retry/repeat the transfer of output files/directories,
- `qcg-refresh_proxy` - refresh the user proxy certificate for the given tasks,
- `qcg-resub` - resubmit the task to be processed once again,
- `qcg-sub` - submit the task to be processes by QCG services.

Resources reservation and control:

- `qcg-rcancel` - cancel reservation(s),
- `qcg-reserve` - reserve resources,
- `qcg-rinfo` - display information about the given reservation(s),
- `qcg-rlist` - list reservations in the system.

System information:

- `qcg-offer` - provides information about current state of resources including their availability and supported applications.

Every task submitted to the system has to be described in a formal way. The default description format - QCG-Simple, is recommended and sufficient for majority of the tasks. The format does not yet allow users to describe more sophisticated scenarios like workflows, parameter sweep tasks, parallel tasks with topologies and these are supported by the XML-based format, called QCG-JobProfile. The QCG-Simple format description file is a plain BASH script annotated with #QCG directives, which is also a common approach for all nowadays queuing systems. The #QCG directives inform the system how to process a task (e.g. define resource requirements and input/output files for running an application). The main difference is that a user has to explicitly specify all files and directories that have to be staged in/out as there is no global shared file system for all sites. Fortunately, staging directives also accept short relative local paths beside the full URLs. Listing 1 presents an example of QCG- SimpleClient job description expressed in the QCG-Simple format. In this example, the NAMD application will be executed with the apoa1/apoa1.namd argument on the hydra cluster with the topology: 12 processes on a single node in the plgrid queue with the walltime limit set to 10 minutes. Prior to the execution, the apoa1.zip file will be staged in and unpacked. After the execution, the whole working directory of the task will be staged out to the results directory, that will be created in the directory from which the task was submitted. Moreover, standard output and error streams will be staged out to the apoa1.output and apoa1.error files, respectively. XMPP notifications concerning status of the task will be sent to tomasz.piontek@plgrid.pl and additionally, every 20 seconds, the application output will be searched for new line containing the ENERGY word which, if present, is sent to the defined mail address.

One of the most frequently requested functionalities that have been recently added to the QCG-SimpleClient is the support for interactive tasks. Depending on particular needs, a user can get an interactive access to the cluster and either run his command line application in the interactive mode or compile their own code and process some test/debugging sessions. The support for this functionality is especially important in the case of systems that do not provide an interactive access at a queuing system level and offer entry only via middleware services.

In the last years we have learned how valuable it is for a user to obtain a detailed status of her/his simulations. Having an access to the data produced only at the end of the job can be accepted in the cases of very short runs only. For this reason, QCG offers a possibility of viewing output of any of its running jobs. Moreover, it is possible to establish an interactive session (using *qcg-connect* command) with an already started batch job. After such an interactive session has been established, a user can inspect the task, for example: list job directory, 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

```
[frame=single]
#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

observing dynamic job metrics displayed in the QCG tools, namely *CPU efficiency* and *memory usage*. Another commonly asked question by end users is *"What time my job will start?"*. The QosCosGrid services attempt to answer this question by extracting this information from the local scheduler. Although this is a best-effort metric, it provides a user with at least a rough estimation of expected waiting time. Finally, what was mentioned in the previous section, QCG enables registering for notifications with application's output, i.e. whenever a given phrase appears in the output file, thus tracking the correctness of a simulation execution.

The QCG-Broker service provides brokering capabilities and, based on information about a current state of the whole system, it can assign a task to the resource in a way that minimizes waiting time in local queues. As an alternative, we provided users with the *qcg-offer* tool. It is a command-line tool that allows regular users to generate queries about free resources available in the grid. The tool, at a QCG-Broker level, leverages the fine-grained information provided by the QCG-Computing services. It is possible to query a single site, to display a full or aggregated view of cluster nodes or to filter resources based on available memory, total/free number of cores, nodes attributes, etc. Listing 2 presents an example output of *qcg-offer*. Users can later utilize this information and their own experience to select a target resource adjusting job size and topology by changing a number of requested nodes and/or slots per node. Moreover, the *qcg-offer* tool is capable of searching for applications and environment modules installed on all sites.

6.2 QCG-Icon

QCG-Icon is a desktop application written specifically for the Windows platform, but also available for Linux and Mac OSX distributions. It was designed

```
[frame=single]
[plgpiontek@qcg ~]$ qcg-offer
HYDRA:
Summary:
    Metric Name        nodes/cores     share
     Total Resources:    279/5252    100%/100%
        Up Resources:    264/4968     94%/94%
      Used Resources:    141/2239     50%/42%
      Free Resources:     82/1432     29%/27% (FreeNodes=2x2,63x12,5x16,11x48,1x64)
  PartFree Resources:    117/2141     41%/40% (AvgFreeCoresPerNode=18)
  Reserved Resources:     34/408      12%/07% (Utilization=0%)

GALERA:
Summary:
    Metric Name        nodes/cores     share
     Total Resources:    194/2688    100%/100%
        Up Resources:    189/2628     97%/97%
      Used Resources:    113/1333     58%/49%
      Free Resources:      0/0         0%/00%
  PartFree Resources:      0/0         0%/00% (AvgFreeCoresPerNode=0)
  Reserved Resources:    151/2172     77%/80% (Utilization=61%)
```

Listing 2: An example of the *qcg-offer* output

to enable an access to selected applications installed on the computing resources of the PL-Grid infrastructure, and is made 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 a large portfolio of applications, including MATLAB, R, NAMD, Gaussian (also integrated with GaussView), GAMESS, Molpro, LAMMPS, Quantum ESPRESSO, Crystal09, NWChem, GROMACS and CPMD. Any other application can also be run as long as a proper BASH script is provided. Despite its simplicity, QCG-Icon delivers most of the functionalities offered by the QosCosGrid stack, including parallel jobs, live output monitoring and providing online statistics about the job resources usage. The overview of the QCG-Icon graphical user interface is shown in Figure 4.

6.3 QCG-Data

The purpose of QCG-Data is provisioning efficient and intuitive synchronization mechanisms for data exchange between a local user file system and the e-Infrastructure. At a low-level, QCG-Data utilizes the iRODS middleware [20] for data storage. The system is directly integrated with QCG-Broker as well as with end user tools and it provides easy management of application input and output files in the QosCosGrid environment. It consists of two layers which make use of iRODS: server layer, which exposes links pointing to data chunks, and the client one, which creates and makes use of those links. The server part is built on the top of the jargon library [29], whilst the desktop application that is integrated with a newest version of QCG-Icon, adapts and extends the iDrop code [28]. The integration carried out between QCG-Icon, QCG-Broker

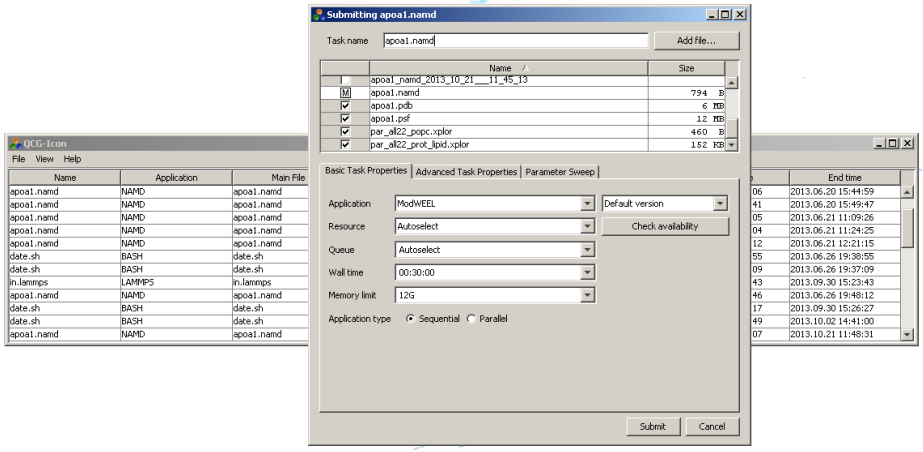

Fig. 4. The "Main" and "New Task" windows of QCG-Icon

and QCG-Data allows users to process jobs which require or produce large data sets.

6.4 QCG-ScienceGateway

The advanced graphic and multimedia-oriented web interfaces designed for scientists and engineers could change the way end users collaborate, deal with advanced simulations, share results and work together to solve challenging problems. With the use of the enhanced version of the Vine Toolkit portal [12] [CROSS-REF to the new article about science gateways], we created the platform called QCG-ScienceGateway. The Gateway consists of a general part displaying and monitoring computational resource characteristics as well as a set of domain-specific web applications developed for certain complex system use cases. Therefore, end users are able to use only web browsers to proceed with their complex simulations with the use of grid and to exchange the results of their studies with co-workers. Currently, QCG-ScienceGateway supports several application scenarios covering such software packages as NAMD, Abinit, QuantumEspresso, NWChem, LAMMPS, nanoMD, SIMPL and Anelli. An example nanotechnology simulation is presented in Figure 5.

7 Integration with EGI Infrastructure

After a successful adoption of QosCosGrid solutions within the Polish research communities, collaboration at the European level has commenced. The efforts resulted in signing the Memorandum of Understanding with EGI (European Grid Infrastructure) in November 2012[2]. This document was an official step towards a sustainable deployment of the QosCosGrid stack into the European grid

[2] https://documents.egi.eu/secure/ShowDocument?docid=1350

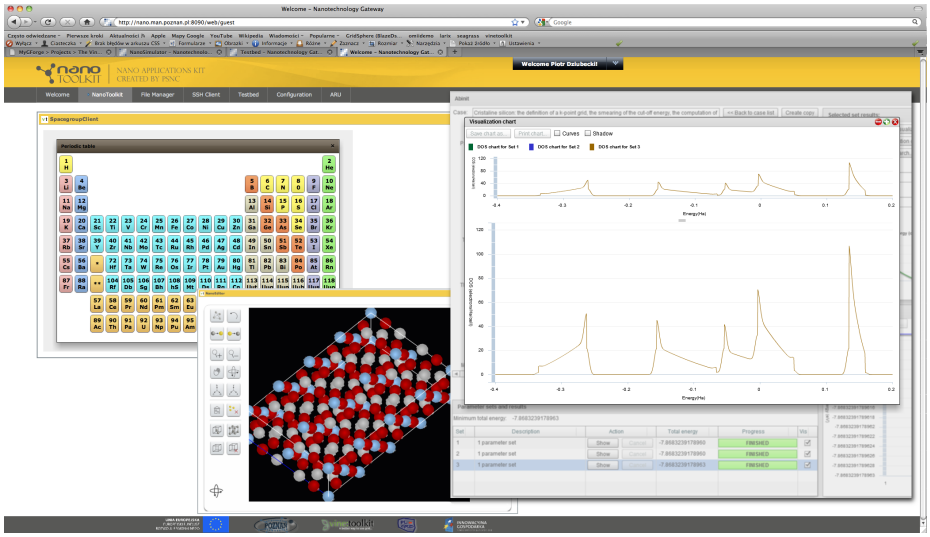

Fig. 5. The sample QCG-ScienceGateway application

ecosystem. The collaboration concerned both: integrating the contributed QCG software components into the operational infrastructure and conducting joint dissemination activities. QosCosGrid services got their own types within EGI and their instances were registered in Grid Configuration Database (GOCDB) — a registry containing general information about the sites and services participating in the production of the European e-Infrastructure. The QCG services were also successfully integrated with the EGI Service Availability System (SAM) and APEL accounting system [15], where they continuously publish requested information. To support European research communities and end users, dedicated QCG support unit was created in the structure of GGUS, that is the EGI helpdesk. In September 2013, after meeting all mandatory requirements and a positive verification of all the core QosCosGrid components, the stack became a part of the Unified Middleware Distribution (UMD) [32]. Similarly, QCG end user tools became available in the EGI Applications Database (EGI-AppDB) [25], which is a repository of tools ready to use within the EGI infrastructure.

Moreover, on the basis of a collaboration between the MAPPER and PRACE projects, the QosCosGrid middleware has been initially validated and accepted for further installation on highly powerful supercomputers available in the PRACE infrastructure. Preliminary installations of basic QosCosGrid services were performed on SuperMUC, HECTOR and Huygens machines.

In August 2013, another 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 objectives of

[3] http://infrastructure.kiev.ua/en/news/114/

this document is to *"provide robust, well-designed, user-centric services to scientific user"* based on the QosCosGrid services. That will be the first QosCosGrid deployment at such a scale outside Poland.

Quite recently, a LCAS/LCMAPS [2] based authorization plug-in has been developed for the QCG-Computing service. This work enabled the QosCosGrid middleware to support authorization mechanism based on the 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 capabilities facilitate the 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[4].

8 Conclusions and Future Work

QosCosGrid is used on daily basis by many researchers in Poland from various research domains, such as quantum chemistry [8], nanotechnology [12], metallurgy, astrophysics and bioinformatics. It is currently the most popular middleware and the first middleware in PL-Grid, taking into account the CPU hours consumed by its users. Tasks controlled by the QCG stack consume, in average, 2 million core-hours per month in total. Moreover, the QCG functions for advance reservations and co-allocation of resources proved to be of particular importance for several complex multi-scale applications developed within the MAPPER project [3][14]. These successes would not be possible without offering simple but powerful end user tools and comprehensive end user support.

What attracts users to the QCG solutions is the fact that the development of QCG tools and services is performed in a close collaboration with groups of domain researchers and is driven by their real needs. The fact that all the aforementioned QCG components are developed by a single group of programmers from Poznan Supercomputing and Networking Center in short development cycles causes that QCG can be adapted to specific requirements in a relatively short time. To the best of our knowledge, QCG currently provides the most efficient and powerful multi-user access to the job management and advance reservation features compared to other existing grid middleware services. QCG also offers unique functionalities and features, such as co-allocation of resources, cross-cluster execution of applications with heterogeneous resource requirements and communication topologies, interactive tasks, as well as asynchronous notifications and monitoring capabilities. In order to meet emerging end user requirements and sophisticated scenarios, QCG provides means by which different e-Infrastructures like EGI, PRACE and EUDAT, as well as the GEANT one in the future, can be bridged.

The continous and sustainable development of QosCosGrid in order to provide highest-quality product for existing and new users, remains a priority of

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

Poznan Supercomputing and Networking Center. In the next few years, we plan to proceed with further deployments of QosCosGrid on Polish and European sites. We want to create extensions and improve current functionalities of the middleware to support new and more demanding computing scenarios. Finally, last but not least, we aim at rendering the use of e-Infrastructure as simple and as user-friendly as possible.

References

1. Alfieri, R., Cecchini, R., Ciaschini, V., dell Agnello, L., Frohner, A., Gianoli, A., Lorentey, K., Spataro, F.: Voms, an authorization system for virtual organizations. In: Grid computing. pp. 33–40. Springer (2004)
2. Alfieri, R., Cecchini, R., Ciaschini, V., Gianoli, A., Spataro, F., Bonnassieux, F., Broadfoot, P., Lowe, G., Cornwall, L., Jensen, J., et al.: Managing dynamic user communities in a grid of autonomous resources. arXiv preprint cs/0306004 (2003)
3. Belgacem, M.B., Chopard, B., Borgdorff, J., Mamonski, M., Rycerz, K., Harezlak, D.: Distributed multiscale computations using the mapper framework. In: Alexandrov, V.N., Lees, M., Krzhizhanovskaya, V.V., Dongarra, J., Sloot, P.M.A. (eds.) ICCS. Procedia Computer Science, vol. 18, pp. 1106–1115. Elsevier (2013)
4. Benedyczak, K., Stolarek, M., Rowicki, R., Kluszczynski, R., Borcz, M., Marczak, G., Filocha, M., Bala, P.: Seamless access to the pl-grid e-infrastructure using unicore middleware. In: Bubak et al. [9], pp. 56–72
5. Borcz, M., Kluszczynski, R., Skonieczna, K., Grzybowski, T., Bala, P.: Processing the biomedical data on the grid using the unicore workflow system. In: Kaklamanis, C., Papatheodorou, T.S., Spirakis, P.G. (eds.) Euro-Par Workshops. Lecture Notes in Computer Science, vol. 7484, pp. 263–272. Springer (2012)
6. Borgdorff, J., Bona-Casas, C., Mamonski, M., Kurowski, K., Piontek, T., Bosak, B., Rycerz, K., Ciepiela, E., Gubala, T., Harezlak, D., Bubak, M., Lorenz, E., Hoekstra, A.G.: A distributed multiscale computation of a tightly coupled model using the multiscale modeling language. In: Ali, H.H., Shi, Y., Khazanchi, D., Lees, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS. Procedia Computer Science, vol. 9, pp. 596–605. Elsevier (2012)
7. Borgdorff, J., Mamonski, M., Bosak, B., Kurowski, K., Belgacem, M.B., Chopard, B., Groen, D., Coveney, P.V., Hoekstra, A.G.: Distributed multiscale computing with muscle 2, the multiscale coupling library and environment. CoRR abs/1311.5740 (2013)
8. Bosak, B., Komasa, J., Kopta, P., Kurowski, K., Mamonski, M., Piontek, T.: New capabilities in qoscosgrid middleware for advanced job management, advance reservation and co-allocation of computing resources - quantum chemistry application use case. In: Bubak et al. [9], pp. 40–55
9. Bubak, M., Szepieniec, T., Wiatr, K. (eds.): Building a National Distributed e-Infrastructure - PL-Grid - Scientific and Technical Achievements, Lecture Notes in Computer Science, vol. 7136. Springer (2012)
10. Ciepiela, E., Nowakowski, P., Kocot, J., Harezlak, D., Gubala, T., Meizner, J., Kasztelnik, M., Bartynski, T., Malawski, M., Bubak, M.: Managing entire lifecycles of e-science applications in the gridspace2 virtual laboratory - from motivation through idea to operable web-accessible environment built on top of pl-grid e-infrastructure. In: Bubak et al. [9], pp. 228–239

11. Demuth, B., Schuller, B., Holl, S., Daivandy, J.M., Giesler, A., Huber, V., Sild, S.: The unicore rich client: Facilitating the automated execution of scientific workflows. In: eScience. pp. 238–245. IEEE Computer Society (2010)
12. Dziubecki, P., Grabowski, P., Krysinski, M., Kuczynski, T., Kurowski, K., Piontek, T., Szejnfeld, D.: Online web-based science gateway for nanotechnology research. In: Bubak et al. [9], pp. 205–216
13. Furlani, J.L.: Modules: Providing a flexible user environment. In: Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V). pp. 141–152 (1991)
14. Groen, D., Borgdorff, J., Bona-Casas, C., Hetherington, J., Nash, R.W., Zasada, S.J., Saverchenko, I., Mamonski, M., Kurowski, K., Bernabeu, M.O., Hoekstra, A.G., Coveney, P.V.: Flexible composition and execution of high performance, high fidelity multiscale biomedical simulations. CoRR abs/1211.2963 (2012)
15. Jiang, M., Novales, C.D.C., Mathieu, G., Casson, J., Rogers, W., Gordon, J.: An apel tool based cpu usage accounting infrastructure for large scale computing grids. In: Data Driven e-Science, pp. 175–186. Springer (2011)
16. Krabbenhöft, H.N., Möller, S., Bayer, D.: Integrating arc grid middleware with taverna workflows. Bioinformatics 24(9), 1221–1222 (2008)
17. Laure, E., Gr, C., Fisher, S., Frohner, A., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., Barroso, M., Buncic, P., Byrom, R., Cornwall, L., Craig, M., Meglio, A.D., Djaoui, A., Giacomini, F., Hahkala, J., Hemmer, F., Hicks, S., Edlund, A., Maraschini, A., Middleton, R., Sgaravatto, M., Steenbakkers, M., Walk, J., Wilson, A.: Programming the Grid with gLite. In: Computational Methods in Science and Technology (2006)
18. MacLaren, J.: Harc: The highly-available resource co-allocator. In: Meersman, R., Tari, Z. (eds.) OTM Conferences (2). Lecture Notes in Computer Science, vol. 4804, pp. 1385–1402. Springer (2007)
19. Palak, B., Wolniewicz, P., Plóciennik, M., Owsiak, M., Zok, T.: User-friendly frameworks for accessing computational resources. In: Bubak et al. [9], pp. 191–204
20. Rajasekar, A., Moore, R., Hou, C.Y., Lee, C.A., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S.Y., Gilbert, L., Tooby, P., Zhu, B.: iRODS Primer: Integrated Rule-Oriented Data System. Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool Publishers (2010)
21. Streit, A., Bala, P., Beck-Ratzka, A., Benedyczak, K., Bergmann, S., Breu, R., Daivandy, J.M., Demuth, B., Eifer, A., Giesler, A., Hagemeier, B., Holl, S., Huber, V., Lamla, N., Mallmann, D., Memon, A.S., Memon, M.S., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Schlauch, T., Schreiber, A., Soddemann, T., Ziegler, W.: Unicore 6 - recent and future advancements. Annales des Télécommunications 65(11-12), 757–762 (2010)
22. Takefusa, A., Nakada, H., Takano, R., Kudoh, T., Tanaka, Y.: Gridars: A grid advanced resource management system framework for intercloud. In: Lambrinoudakis, C., Rizomiliotis, P., Wlodarczyk, T.W. (eds.) CloudCom. pp. 705–710. IEEE (2011)
23. Troger, P., Rajic, H., Haas, A., Domagalski, P.: Standardization of an API for Distributed Resource Management Systems. In: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid. pp. 619–626. CCGRID '07, IEEE Computer Society, Washington, DC, USA (2007), `http://dx.doi.org/10.1109/CCGRID.2007.109`
24. Distributed Resource Management Application API Version 2 (DRMAA), `http://www.ogf.org/documents/GFD.194.pdf`

25. EGI Application Database (AppDB), http://appdb.egi.eu/
26. Grid-SAFE accounting framework, http://gridsafe.sourceforge.net
27. HPC Basic Profile Version 1.0, http://www.ogf.org/documents/GFD.114.pdf
28. iDrop - the client tool for iRODS, https://code.renci.org/gf/project/irodsidrop
29. Jargon library for iRODS, https://code.renci.org/gf/project/jargon
30. OASIS Web Services Notification, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn
31. SAGA project, http://saga-project.org
32. Unified Middleware Distribution (UMD), http://repository.egi.eu/