Sun Grid Engine, a new scheduler for EGEE middleware

G. Borges[1], M. David[1], J. Gomes[1], J. Lopez[2], P. Rey[2], A. Simon[2], C. Fernandez[2], D. Kant[3], K. M. Sephton[4]

[1] Laboratório de Instrumentação em Física Experimental de Partículas Lisboa, Portugal
[2] Fundación Centro Tecnolóxico de Supercomputación de Galicia (CESGA), Santiago de Compostela, Spain
[3] CCLRC, Rutherford Appleton Laboratory, Oxfordshire, UK
[4] London e-Science Centre (LeSC), Imperial College London, UK

Abstract. The Enabling Grids for E-SciencE (EGEE) is the world largest operating grid infrastructure in place serving thousands of world wide multi-science users with robust, reliable and secure grid services. In order to integrate a large number of institutions, often with different setups and configurations, the EGEE middleware, gLite, must cope with a broad set of local requirements, such as the site Local Resource Management System (LRMS). This paper presents the work developed to integrate Sun Grid Engine (SGE), a Sun open source resource management software, with the EGEE middleware, allowing to increase the gLite offer, and easing the integration of new clusters in EGEE infrastructure.

1 The EGEE infrastructure

The Enabling Grids for E-Science [1] is a partially funded European project in the framework of the 6th Information Society Technology (IST) programme. Its fundamental goal is the deployment of a Pan-European grid infrastructure for all fields of science, where the resources available to general users are supported by a set of agreed services provided and supported by the EGEE community.

EGEE grid services are divided in two different sets: The *local services*, deployed and maintained by each participating site, and the *core services*, installed only in some Resource Centres (RCs) but used by all users. Two of the main components of the grid core services are the Resource Broker (RB) and the Top-Berkeley-Database Information Index (top-BDII). The RB is of most importance since it interacts with almost every other low level service. It implements the so called matchmaking process, that is, it will pass the job submission requests to the most appropriate site for execution, matching the specific requirements expressed by the user to the available resources at submission time. The RB decision regarding which site is chosen for the job execution is performed taking into account the grid resources information published through the top-BDII. The top-BDII collects information provided by the Grid Index Information System

(GIIS) running at each site (ex: Free slots, free storage capacity, etc) and stores it in a single database, updated every couple of minutes.

A typical EGEE grid site must be equipped with the necessary resources allowing an uniform and transparent access of a grid user to the global infrastructure. At the same time, it must deploy services for the inter-connectivity with the core services and other sites. A local grid site is basically composed by a Computing Element (CE), a site Berkeley-Database Information Index (site BDII), a Storage Element (SE), a Monitoring Box (MonBox) and a User Interface (UI). The Computing Element (CE) is a key element of the site. It runs a "gatekeeper" service which executes authentication and authorization mechanisms allowing grid users to use the site resources, acting as an interface to the local batch system and enabling them to submit jobs, query their status and retrieve the output results. The job execution is performed in one or several worker nodes and the results are retrieved back to the user.

This schema is now commonly accepted and used worldwide. EGEE presently interconnects a large number of sites in more than 32 countries and integrates several national and regional grids which, although using their own tools, have agreed to install and deploy the same middleware for a fast integration and cooperation in European grid operations.

2 The gLite middleware

A robust and reliable grid infrastructure can not be achieved through the implementation of a set of disconnected components. The grid middleware is the software layer which glues together all local and central services allowing a transparent and uniform interoperability within the infrastructure. EGEE also addresses the development and hardening of previous grid middlewares, as the ones coming from the European Data Grid (EDG) and LHC Computing Grid (LCG) projects [2]. In this sense, EGEE has created gLite [3], proposed as the appropriate middleware to support the necessary grid services for multi-science applications. It relies on existing software packages as the Globus Toolkit (GT) [4], Condor [5] and on the Virtual Data Toolkit (VDT) [6] and follows a Service Oriented Architecture (SOA) which facilitates interoperability among Grid services, allows easier compliance with upcoming standards and permits that services work together in a concerted way being also able to be used independently. Such architecture is presently based in five different building blocks:

- Grid access services, which provide a common framework by which the user may gain access to the grid resources.
- Security services, providing auditing protocols for a dynamic and proportional response to security incidents.
- Information and monitoring services, which provide a mechanism to publish information regarding grid resources and use it for monitoring purposes.
- Job management services, dealing with job execution, workload management, accounting and job provenance.
- Data services, enabling data storage using application metadata catalogues.

The middleware is supported mainly for Scientific Linux CERN version 3 distribution although a huge effort is now being carried out to port it to Scientific Linux CERN version 4 and to other operating systems such as SUN Solaris.

3 SGE and others Local Resource Management Systems

The Local Resource Management System (LRMS) is the generic denomination for the cluster component which manages the execution of user applications. gLite must be able to interface and inter-operate with a wide set of LRMS, in order to easy the integration of already operating clusters wishing to join EGEE.

Presently, there is a large number of available LRMS, which are often chosen according to their ability to optimize the cluster resource usage, enable and fulfill a broad range of usage policies and easy the cluster administration tasks. gLite, by default, deploys and supports two different kinds of LRMS, the Load Sharing Facility (LSF), widely used at CERN, and the Tera-scale Open-source Resource and QUEue manager (Torque) used together with Maui. One can also find other LRMS within EGEE infrastructure, as Condor and SGE, for which there is limited or non-existent support up to now. From the 264 EGEE CEs, 220 use Torque, 22 use LSF, 8 use Condor and 8 are configured with SGE.

LSF [7] is an extremely advanced system developed by Platform Computing Corporation. It contains all the requirements a LRMS should have, as flexible job scheduling policies, advance resource management (check pointing, job migration, load balancing), graphic interfaces to monitor cluster functionalities and it is integrable in Grid infrastructures. The big disadvantage is that it is an expensive commercial product addressed to big computing facilities and not suitable for small computing clusters.

The Open Portable Batch System (OpenPBS) [8] is an open source software developed for academic use but discontinued by Altair which devoted all their efforts to PBSPro, their commercial closed source version. Further developments on OpenPBS were latter addressed by Cluster Resources Inc. and renamed as Tera-scale Open-source Resource and QUEue manager (Torque) [10]. The big advantage of this package is its interface and integration with parallel libraries which allows to start parallel processes through PBS services. The system has full control of the parallel instances (tight integration) and can properly manage accounting information regarding resource usage. Its biggest problem is that the configuration of all components (queues, policies, etc...) is done from the command line. The graphical interface for job monitoring is also not very user friendly. Moreover, the Torque software development has stopped meaning that the future of the project is uncertain.

Maui [11], when integrated with other LRMS, is normally used as the primary scheduler but it can be also used independently. It can work together with OpenPBS, PBSPro, Torque, SGE, LSF, and is well suited for implementing fair share policies, back-filling, resource reservations and assigning priorities based on job characteristics. Its main disadvantage is that job scheduling is only fully efficient if users properly describe their requests.

Finally, Condor [5] is a job management system developed for High Throughput Computing (HTC). In HTC environments, high efficiency is not playing a major role since the computing needs are basically fulfilled by CPU and memory usage. The main challenge a typical HTC cluster faces is how to maximize the amount of resources accessible to its users. The basic advantages of Condor are the CPU harvesting, the special ClassAds language for job and client description and dynamic check-pointing and migration. Condor should be easily integrated with Grid systems since it contains mechanisms for authentication and interface with Globus. Its disadvantage is that it is not optimal for parallel applications and check-pointing only works for serial jobs.

3.1 Why Sun Grid Engine? Advantages and Disadvantages

Sun Grid Engine (SGE) [12] is an open source job management system supported by Sun Microsystems and used worldwide for distributed resource management and software/hardware optimization in heterogeneous networked environments. SGE can increase machines and application licenses productivity at the same time as optimizes the number of jobs that can be executed and completed.

When a user submits a job for execution, it is first sent to a scheduler where its requirements and priorities are determined. Afterwords, a master daemon is notified and the job is assigned to a holding area where it waits for execution. The queues are located in server nodes and have attributes which characterize the properties of the different servers. The user may request at submission time certain execution features (memory, execution speed, available software licenses) and the job will only be executed in queues (execution servers) which are able to provide the corresponding service. One big SGE advantages is the fact that site administrators can implement new server features, programming their own load sensors and assigning them to precise execution queues. When a queue (server) becomes available for execution, SGE determines which jobs may run on that queue and immediately dispatches the one with the highest priority or the longest waiting time. An executive daemon starts the job from the queue, monitors its progress, reports load, memory consumptions (and other configurable features) to the scheduler and updates the master daemon at finish time.

SGE supports check-pointing and migration but you may need to configure a check-pointing interface and rely on additional programing to invoke external check-pointing commands. SGE also implements calendars used to define periods in which execution queues may or may not be available. This utility could be useful to run short time jobs in fluctuating resources resulting in an automatic and dynamic extension of the cluster. Finally, one of the most important advantages of SGE is its intuitive graphic interface which can be used by users to submit jobs and by site administrators to configure and manage their cluster, being able to access almost all system functionalities in a friendly way. Site managers can also save configuration templates that they can upload at any time.

Obviously, there are disadvantages too. The SGE Scheduler module presents limitations defining scheduling policies and one has to think in complicated mechanisms for pre-emptive scheduling and making reservations. Other schedulers, as

OpenPBS and Torque, offer better support for parallel tasks through the provision of a library that applications can call for spawning remote processes and that many MPI implementations support. Nevertheless, tight integration within SGE is supported through an SGE specific version of "rsh" called "qrsh". Processes started using "qrsh" are tracked by SGE, killed when the job finishes and included in SGE's accounting statistics. As the standard remote execution model for many parallel libraries/applications involves calling "rsh" (or "ssh"), tight integration support for SGE is often fairly trivial to setup and unlike OpenPBS and Torque, may not require any changes to the parallel libraries/applications themselves, though there are some complications that can prevent a trivial approach to tight integration in SGE from working:

- SGE will normally limit the number of simultaneous copies of "qrsh" according to how many slots were allocated on each node. This made support for the LAM MPI [13] library slightly more complicated since LAM MPI uses multiple calls to "rsh" to setup and shutdown their own daemons.
- Some parallel applications/libraries deliberately try to disassociate the processes they create from their parent process group causing problems for accounting tracking. This is also seen with single processor jobs regardless of the underlying queuing system. There is a SGE feature specifically designed to address this issue by using unix groups and process groups to track jobs.

SGE is still a "work in progress" project meaning that the observed flaws may be addressed to dedicated teams and support is assured by dedicated staff.

4 Sun Grid Engine and gLite integration

The Computing Element is the grid component withing gLite middleware responsible for accepting grid batch jobs and grid job control requests, mapping grid tasks to local user accounts through the Local Credential MAPping Service (LCMAPS) [14]. Presently, there are three different gatekeeper flavours:

- The lcg-CE, based on GT2 GRAM and on GSI-enabled Condor
- The gLite-CE, based on GSI-enabled Condor-C
- CREAM [15], a Web Services based interface

In the first phase, we have focused on the SGE integration in the lcg-CE. The Grid Resource Allocation and Management (GRAM) service passes the job to a layer which interacts with the LRMS, called the JobManager. The JobManager function is the translation between the general-purpose gatekeeper requests and the software-specific interface of the underlying cluster. Within the most basic functionalities, the JobManager must be able to receive the grid query (job submission, job canceling, get job status, etc ...), translate it to the site local scheduler and retrieve the site local scheduler response back to the grid user.

The CE also runs a Grid Resource Information System (GRIS). The GRIS service is responsible for extracting the current state information from the underlying cluster system (ex: number of available CPUs, their type and performance,

operating system, installed applications, etc...) and feed it, through various processes, to the Grid Index Information System (GIIS) running on the site-local BDII host which acknowledges it to the outside world.

Sun Grid Engine is already used in EGEE although not officially supported within the gLite framework. EGEE sites supporting SGE enable interoperability with gLite middleware through plugins developed at Imperial and CESGA, and deployed on a disconnected and best effort basis. In the absence of a standard way to implement such LRMS within EGEE, site managers from CESGA, LeSC and LIP have focused on a community effort to provide official gLite support for SGE [16], namely for the lcg-CE gatekeeper deployment. To achieve a general and common way to install, deploy and configure a SGE grid site within EGEE infrastructure, four different items were addressed (see also Figure 1):

- Adapt the JobManager to translate between Grid gatekeeper requests and SGE software enabling job submission, job canceling or retrieve job status.
- Produce the appropriate tools to parse the SGE accounting log file and produce correct accounting information.
- Change and configure the CE GRIS in order to properly query SGE and extract the cluster current state information, and interface it with the GIIS.
- Adapt the standard EGEE installation and configuration tool (YAIM) to properly setup a Grid site running SGE.

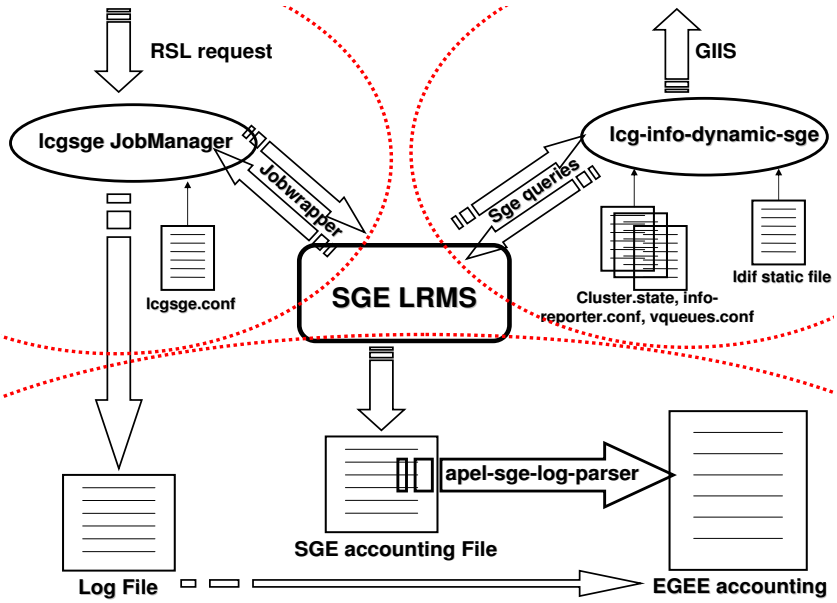

Fig. 1. SGE integration schema in lcg-CE gLite module.

4.1 Site JobManager

The JobManager is the core component of the Globus GRAM service. It takes care of submitting the jobs to the LRMS based on the requests it received using the RSL language [17]. This it done creating a script called *jobwrapper* that is then submitted to the LRMS. The JobManager is also used as an intermediary to query the status of the jobs and to cancel them.

The implementation of the SGE JobManager [18] presented in this work is based on the LCG PBS JobManager although other implementation based on the Globus JobManager is also available [19]. It requires the XML::Simple.pm perl module. The SGE client tools (qsub, qstat, etc) should be available in the CE, even if the SGE qmaster is installed in another machine. It does not require share directories but the home directories should have the same path in the CE and in the WN. This is the common configuration in most EGEE sites where each WN has its own home directories but using the same path as in the CE.

The SGE JobManager has re-implemented the following functions to be compatible with SGE:

- *submit*: Checks the RSL arguments returning a Globus error in case the arguments are not valid or if there are no resources. If no errors are returned, a *job_id* is assigned.
- *submit_to_batch_system*: Submits jobs to the batch system, after building the *jobwrapper* script, to be submitted to the local SGE batch system, by getting the necessary information from the variables filled with the RSL arguments.
- *poll*: Links the present status of jobs running in SGE with the Globus appropriate message.
- *poll_batch_system*: Allows to know the status of jobs running in the site cluster parsing the output of the *qstat* SGE command.
- *cancel_in_batch_system*: Cancels jobs running in SGE using *qdel*.

4.2 Accounting Information (APEL)

APEL is a log processing application which is used to interpret gatekeeper and batch system logs to produce CPU job accounting records. APEL's implementation is based on a plug-in architecture which separates the core functionality from the actual log parsing.

APEL requires the JobManager to add "gridinfo" records in the log file. Standard Globus JobManagers do not log them but LCG JobManagers implemented this functionality. Since the SGE JobManager implements the interface of an LCG JobManager it already provides this information.

To add support for SGE it was necessary to develop a plug-in (*apel-sge-log-parser*) to parse the SGE accounting log file. This information, together with the gridinfo mappings from the JobManager are joined together to form accounting records that are then published using R-GMA to an accounting database.

4.3 Site Grid Index Information Service (GIIS)

The key part of GIIS that involves SGE LRMS is an information plugin script that is run every few minutes and examines the state of the queuing system reporting back details such as the number of pending and running jobs and estimated response times for each queue based on the current state of the system.

Some work has been done by the authors of the OpenPBS/Torque plugin to generalise their version of the information provider script so that new batch systems could be supported by simply adding what was assumed would be a trivial script to convert the output of the queue listing command into a standardised format. However there are some significant discrepancies between the generic model that was assumed and that of SGE...

- SGE was originally designed such that users submitting jobs were expected to specify the requirements of their jobs rather than the queues under which jobs would subsequently be run. Unless explicitly specified, SGE does not assign a job to a queue until execution time. What SGE originally referred to as a "queue" is perhaps more constant with that of an "execution slot". Version 6.0 of SGE introduced a concept of "cluster queues" which can be used in a similar way to queues in other batch systems, however not all existing SGE sites that have local as well as grid users will want to operate a policy of having users specify a cluster queue when submitting jobs.
  As the information expected to be reported by the GIIS is based on queues, a concept of "virtual queues" is used. In the simplest case, each virtual queue simply maps directly onto a cluster queue, however mappings onto more complex sets of constraints are also possible (for example LeSC uses virtual queues based on differing time limits).
- Not all of the fields expected to be available for use by the generic information provider are present in the output of SGE's qstat command. Specifically, SGE only reports the time a job last changed state rather than both the submission and start time of each job.
- SGE has a concept of task arrays that cannot efficiently be represented using the generic queue model. Task arrays allow a single batch script to be executed multiple times with the only difference between each execution instance being an environment variable that is set to indicate which instance of the task array is being run. Although task arrays cannot be submitted via glite, such arrays may still be present within the queuing systems at sites that have local users. By calculating job counts and other information directly, we can simply factor in the number of tasks within a task array, rather than having to iterate through what may be many thousands of individual tasks.

Because of these issues, the solution implemented for SGE does not currently use the generic script. Instead, it uses a standalone script loosely modelled on the generic plugin and on the LSF plugin. Its current implementation is based on previous versions of the script developed at Imperial College London [19].

In order to generate its output, the information reporter reads in a copy of a static ldif file generated when the site is originally configured and containing

details of all of the "virtual queues" that are to be reported on (plus default values to be used in the event of a problem with the information provider). It also reads in a configuration file specifying how each virtual queue maps into a list of resource requirements. Most of the dynamic information required by the information reporter is determined from a single call to SGE's "qstat" command. For reliability, "qstat" is called with the "-xml" argument in order to ensure that the output is in a machine readable format and will not be misparsed. Other arguments passed to qstat are to request the detailed status of each queue instance on the system (including the resources they offer), and the details of each job including the resources it requires.

For each queue instance (execution slot) and job that is found in the qstat output, the system has to determine which virtual queues the execution slot or job should be associated with. Once this mapping to virtual queues has been determined, each virtual queue is then considered in turn to count up the number of job slots (processors) it can utilise, the number of pending and running jobs and the total amount of runtime left on all of the jobs assuming that they will run for their maximum duration. This information is then used to calculate estimated response times for newly submitted jobs using a similar approach to the information providers developed for other batch systems.

One issue that arose whilst developing the information plugin, was that on a live system, the state of the batch queues can change quite rapidly, making it difficult to analyse the effect of changes to the code and track down problems. To solve this issue, support was built allowing to capture a copy of all information provider input data (the current time, the static ldif file, the configuration files and the output of the various SGE commands). This data can be replayed to the information provider using dummy versions of the SGE commands.

4.4 YAIM integration

YAIM (Yet Another Installation Method) [20] is a gLite tool to install and configure grid services. Its aim is to provide a simple installation and configuration method for small and medium grid sites but it can also be extended and applied for larger clusters. Indeed, it separates the installation process from the configuration one, meaning that big sites can install EGEE middleware using a local fabric management system and then configure it using YAIM.

The installation of the gLite middleware is done using standard apt tools although yum is also supported. For any given node type (CE, SE, RB, etc...) there is a *meta rpm package* which does not contain any software but depends on a large set of other packages. In this way, the installation of a single rpm automatically pulls out the necessary software from the repository.

YAIM is based on a library of bash functions called by a configuration script to set up the appropriate grid nodes. The topology of the grid site is totally encapsulated on a file called site-info.def containing the site definitions, with a site-wide validity. The different functions called to configure each node type are defined on the node-info.def file.

An official support aiming to include SGE in gLite middleware has to automatically go through YAIM tool to deploy, install and configure SGE in the different grid nodes. SGE is not included in the EGEE repositories, and therefore, we have compiled SGE Version 6.0, Release 7.1, under SLC4 although it can also be used in SLC3. The main difference between both operating systems is that the SGE Berkeley Data Base Spooling requires version 4.2 of the libdb library which is only available in SLC4. Therefore, to use SGE software under SLC3, it requires the additional packaging of libdb-4.2.so.

The SGE installation and configuration addressed in this report relies on a Qmaster machine (the scheduler head node) installed in the CE gatekeeper, and on SGE execution machines running in the WNs. The previous rpms will deploy all software under /usr/local/sge/V60u7_1 and link this directory to /usr/local/sge/pro allowing to keep track of old SGE versions and use them when needed. $SGE_ROOT environment variable, pointing to the SGE installation directory, must then be set to /usr/local/sge/pro.

Two new nodes were defined in the node-info.def file, the *CE_sge* and the *WN_sge*. The functions run at configuration time include the same set as for the standard *CE* and *WN* nodes with two additional ones, *config_sge_server* and *config_sge_client*, respectively for the *CE_sge* and *WN_sge*.

The *config_sge_server* takes care of the whole configuration of the SGE Qmaster server. It requires 3 new variables which have to be previously defined in the site-info.def file: SGE_QMASTER (which at present time can only be set to the CE host), DEFAULT_DOMAIN (the site default domain) and ADMIN_MAIL (the SGE administration email). A perl script *(configure_sgeserver.pm)* builds all the default SGE directory structure, namely $sge_root/default/common/ and $sge_root/default/spool, and configures important default files needed to run SGE, as the environment setting files, the global configuration file, the scheduler configuration file, and the complex variable list. Finally, the perl script updates the runlevel information of the sge daemons (sge_qmaster) and declares them in /etc/services. The Qmaster service is then safely started.

The *config_sge_server* script continues to address the specific configurations requested to process EGEE jobs. It defines one execution queue (including all execution hosts) for each VO and their permission lists (userset lists). For this operation, it is important that the WN_LIST, USERS_CONF, VOS and QUEUES variables are properly defined in the site-info.def file since they are used to build the SGE exec node list, the SGE userset lists and the SGE local queues. The *config_sge_server* bash function also builds the JobManager configuration files, configures Globus to use the *lcgsge* JobManager, and implements the SGE information reporter and virtual queue configuration.

On the other hand, the client configuration in the WNs is much simpler and implemented by the *(config_sge_client)* function. The philosophy is the same as the one implemented in the CE gatekeeper: A perl script *(configure_sgeserver.pm)* deploys the SGE configuration files for the execution nodes and the corresponding execution daemons (sge_execd). Here, it is important to configure $sge_root/default/common/act_qmaster file which identifies the Qmaster host.

In summary, all the discussed YAIM changes are implemented installing the lcgCE-yaimtosge-0.0.0-2.i386.rpm and gliteWN-yaimtosge-0.0.0-2.i386.rpm packages, which may be downloaded from [21]. These new packages can only be installed if the lcg-CE and glite-WN meta packages are already present. Apart from that software, they also require the SGE rpms, glite-yaim (>= 3.0.0-34), perl-XML-Simple (>= 2.14-2.2), openmotif (>= 2.2.3-5) and xorg-x11-xauth (>= 6.8.2-1). From that point, it should be possible to configure a SGE Computing Element running the YAIM tool in the standard way, but referring to the new *CE_sge* and *WN_sge* hosts.

5 Conclusions and Future work

In this paper we have presented the work developed to integrate SGE, a Sun open source job management system, within gLite middleware. Structural changes to fulfill this integration were addressed in four different local middleware elements: the site JobManager, the accounting tools, the site GIIS and the YAIM installation and configuration software. A lcg-CE gLite component running SGE as LRMS can now be installed and configured using the standard EGEE tools.

Future work is likely to include enhancements to the information provider to further improve its flexibility and take into account overlapping cluster queues, and overlapping virtual queue definitions. The possibility of modifying the script to output the virtual queues in a form that can be used by the generic information provider is also being considered so that the output of the two methods can be compared. One argument given in favour of the generic information provider by its authors is that it ensures that the estimated waiting times are calculated in a consistent manner between different queuing systems in order to avoid any unwanted bias... though it is likely that the generic calculations will offer a less accurate reflection of the state of a particular batch system as they cannot take into account the specific characteristics of each system.

The YAIM changes will addressed further developments in order to separate the SGE Qmaster host from the Computing Element and to allow more dynamic and flexible configuration procedures.

Finally, we also intend to write GridIce sensor scripts, a tool which is still not fit to be integrable with SGE.

Work has started on integrating support for BLAHP, running on the glite flavour of the CE, which is intended to take over the role of the existing job managers within the glite software stack that were originally derived from Globus. It is expected that the BLAHP implementation will share the configuration files and concept of virtual queues with the information provider. The other local middleware elements (GIIS, YAIM) basically remain unchanged for this CE flavour.

6 Acknowledgements

We would like to acknowledge to David McBride, who wrote the original version of the information provider that was subsequently enhanced, to Juan Fontan for his original SGE JobManager development, and to João Paulo Martins and Francisco Barnabé for their contributions to this paper.

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org.

References

1. http://www.eu-egee.org/
2. http://eu-datagrid.web.cern.ch/eu-datagrid/; http://lcg.web.cern.ch/LCG/
3. http://glite.org/
4. http://www.globus.org/
5. http://www.cs.wisc.edu/condor/
6. http://www.cs.wisc.edu/vdt/
7. http://accl.grc.nasa.gov/job_schedulers/lsf/index.html
8. http://www.openpbs.org/main.html
9. http://www.altair.com/software/pbspro.htm
10. http://www.clusterresources.com/pages/products/torque-resource-manager.php
11. http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php
12. http://gridengine.sunsource.net/
13. http://www.lam-mpi.org/
14. http://www.dutchgrid.nl/DataGrid/wp4/lcmaps/
15. http://grid.pd.infn.it/cream/field.php
16. https://twiki.cern.ch/twiki/bin/view/LCG/GenericInstallGuide301; https://twiki.cern.ch/twiki/bin/view/LCG/ImplementationOfSGE
17. http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html
18. http://www.egee.cesga.es/lcgsge/releases/0.22/
19. http://pubs.doc.ic.ac.uk/lesc-grid-site/; http://www.lesc.ic.ac.uk/projects/SGE-LCG.html
20. https://twiki.cern.ch/twiki/bin/view/EGEE/YAIM; https://twiki.cern.ch/twiki/bin/view/EGEE/YaimDevelopersGuide
21. http://www.lip.pt/grid/