Reservations for Compute Resources in Federated e-Infrastructure

Marcin Radecki[1], Tadeusz Szymocha[1], Tomasz Piontek[2], Bartosz Bosak[2], Mariusz Mamoński†[2], Paweł Wolniewicz[2], Krzysztof Benedyczak[3], and Rafał Kluszczyński[3]

[1] AGH University of Science and Technology, ACC Cyfronet AGH, ul. Nawojki 11, 30-950 Kraków, Poland
tadeusz.szymocha@cyfronet.pl

[2] Poznan Supercomputing and Networking Center, Institute of Bioorganic Chemistry of the Polish Academy of Sciences, ul. Noskowskiego 12/14, 61-704 Poznań, Poland

[3] University of Warsaw, Interdisciplinary Center for Mathematical and Computational Modelling, ul. Pawińskiego 5a, 02-106 Warszawa, Poland

Abstract. This paper presents work done to prepare compute resource reservations in the PL-Grid Infrastructure. A compute resource reservation allows a user to allocate some fraction of resources for exclusive access, when reservation is prepared. That way the user is able to run his/her job without waiting for allocating resources in a batch system.

In the PL-Grid Infrastructure reservations can be allocated up to amount negotiated in a PL-Grid grant. One way of getting reservation is allocation by a resource administrator. Another way is to use predefined pool of resources accessible by various middleware. In both approaches once obtained, reservations identifiers can be used by middleware during job submissions. Enabling reservations requires changes in middleware. The modifications needed in each middleware will be described. The possible extension of existing reservation model in the PL-Grid Infrastructure can be envisaged: reservation usage normalization and reservation accounting.

The reservations are created and utilized in the user's context, so there must be a way to pass the reservation details from the user-level tools to a batch system. Each of PL-Grid supported middleware, namely gLite, UNICORE and QosCosGrid, required adaptations to implement this goal.

Keywords: QosCosGrid, UNICORE, gLite, advance reservations, co-allocation of resources.

1 Introduction

The scheduling of computing jobs at HPC clusters may be done in different ways according to the resource provider policy:

- *advance reservations*, when the user needs resources (limited to a single system, location) for a specific time span in future,

M. Bubak et al. (Eds.): PLGrid Plus, LNCS 8500, pp. 80–93, 2014.

- *co-scheduling*, when simultaneous access to resources that are part of different logical systems (or locations) is required,
- *on-demand scheduling*, when jobs for a time-crucial application should be run as quickly as possible; the existing jobs are let to finish or running jobs may be preempted to allow a new job to start,
- *workflows*, when jobs have dependencies between them and one job is waiting for output of the other job (e.g. a visualisation job should start after core computations have finished).

The scientists usually need computational resources to perform complex theoretical calculations. In most cases, typical submission of a job to the queuing system is sufficient. The computations are done on shared resources at not exactly predictable time in future. But for some use cases it may be required to run jobs without waiting in a batch system queue, at exclusive resource or/and at specific time in future.

The PL-Grid Infrastructure is supporting *advance reservations*, *workflows* and to a limited extent *co-scheduling*. The QCG middleware has pre-agreed pool of reserved resources that provide the users with the global *co-scheduling* mechanism based on advance reservations at every PL-Grid Infrastructure computer center.

The *advance reservations* are "expensive" meaning that a reserved resource cannot be used by other users and if it is not occupied by jobs, it may be considered as not optimally used. For this reason, the accounting of reservations needs to be in place to efficiently manage a user's request. Thus, the *advance reservations* are covered by the PL-Grid grant system. The user may apply for a grant, in which he declares that the advance reservation will be needed and specifies the general values for size, time-span and recurrence of the reservation. Then, while the grant is active, the user can apply for setting the reservation in advance, which is then manually set by a system administrator. The user is informed about a reservation identifier and can specify it in his own scripts or middleware tools. There is also some cost incurred by freeing resources needed for the reservation. This is more substantial for large reservations and is similar to freeing resources for executing a many-CPU job, however, in the PL-Grid Infrastructure we decided to neglect this kind of cost.

2 Related Work

The advance reservation is of interest to users of different existing IT infrastructures. The need of advance reservation was tackled also by I. Foster et al. and yielded in proposition of the Globus Architecture for Reservation and Allocation (GARA) [1].

Parallel to the European infrastructure, the United States infrastructure was developed within the TeraGrid and its continuator – the Extreme Science and Engineering Digital Environment (XSEDE) [2] project. The metascheduling Requirements Analysis Team (RAT) recommended evaluation of tools facilitating

reservations management [3]. However, authors were not able to reach public documents describing technical implementation. The RAT indicated the Highly-Available Resource Co-allocator (HARC) [4] as recommended for further evaluation in the TeraGrid project. The goal of HARC is to provide co-allocation service for metacomputing and workflow applications, where diferent types of resources are reserved as they were a single, indivisible resource. So, the allocation process reminds database transaction.

An interesting approach of dealing with the issue of resource reservation at a middleware layer is use of a *pilot jobs* framework as presented in [5]. A pilot job occupies a resource and pulls down some other job to be executed at this resource. This way it is possible to avoid interaction with batch system mechanisms for resource reservation, which may be troublesome, having in mind a variety of batch systems in grid environment, but currently (2014) it is not used in the PL-Grid Infrastructure.

3 The Grid Resource Allocation and Agreement Protocol

The Grid Resource Allocation and Agreement Protocol working group (GRAAP-WG) [6] of the Global Grid Forum (GGF) defines methods and means to establish Service Level Agreements between different entities in a distributed environment. The advance reservation is one of the defined scenarios.

GRAAP-WG defines the advance reservations scenario as to perform a job submission within the context of an existing (or advance) reservation of capability or pre-established resource preferences. The difference from the simple job submission is that the user knows that he has an ongoing relationship with the job hosting service, and can expect his job offer(s) to be accepted as long as the requested parameters are kept within certain limits set by the relationship. For example, the reservation might guarantee availability of a certain kind of resource on a certain schedule, or with a particular cost model. Reservation is an abstraction for understanding this refined expectation about the handling of future jobs; whether the job hosting service uses preemption, predictive models, or the literal setting aside of resources is an implementation decision for the service. Another use of pre-established agreement is to specify resource preferences, e.g., choice of nodes with a certain amount of memory over others, via an agreement, that are to be used in all subsequent resource allocations to incoming jobs in the context of this agreement.

GRAAP-WG proposes an architecture comprising two main layers: the Agreement layer and the service provider layer. The Agreement layer implements the communication protocol used to exchange information about Service Level Agreements and defines its specification. The reservation and allocation request is issued by the agreement initiator. The Agreement layer is responsible for ensuring that the guarantees defined in the "contract" – the Agreement – are enforced by a suitable service provider. In addition, the Agreement layer defines the mechanisms:

– to expose information about types of a service and the related agreement offered (the Agreement templates),

– to handle the submission of service requests (the so-called agreement offers) and submit them to the Agreement layer; one offer needs to comply with at least one template exposed by the Agreement provider, to which it is dispatched, and it has to meet the agreement creation constraints specified in the corresponding Agreement template.

The Agreement layer relies on service providers. An Agreement is successfully created if one or more service providers are able to enforce the guarantees associated with it. The service provider is responsible for supervising the status of a pool of resources and enforcing the agreed guarantees associated with them. Each Agreement Factory can interact with one or more service providers. The actual enforcement mechanisms supported by a given service provider, depend on the type of technology the provider supports.

Generally speaking, not all the resource instances in one grid infrastructure need to support service providers for reservation and allocation. The possibility to do a reservation and an allocation depends on the type of technology the resource is based on.

4 Results

The reservations in the PL-Grid Infrastructure are delivered based on the following policy. In order to allocate a reservation on resources, the request for it should be registered in the PL-Grid grant system. Then, after the PL-Grid grant is active, a user can apply for preparation of reservation. The user specifies number of reserved slots and time of reservation (*Max. res. total walltime*). The agreed reservation allows the user to request the reservation up to the limits defined in the PL-Grid grant system, during application for the grant. There are restrictions in delivery of reservations due to the local resource provider policy: for example, minimum number of slots required, minimal walltime required or period of inactivity of the user. The reservation can be cancelled, but the user is charged for the time the reservation was active. The reservation time is accounted as if the resources were used for all the declared time by the user.

4.1 Reservations in gLite

Using LRMS and VOMS Based Reservations with gLite. The scheduling methods mentioned in Introduction form different requirements for service providers and resource managements systems.

Co-allocation is not supported in gLite, although there is a proposed extension to gLite architecture (see below).

Exclusive access to resources can be guaranteed by specifying in Job Description Language (JDL) job description file attribute *WholeNodes=yes*. The WholeNodes attribute indicates whether whole nodes should be used exclusively or not.

Most LRMS support advance reservation, but even if a user or a Virtual Organization (VO) manually agrees reservation with sites, it is impossible to

request a specific reservation identifier (ID) in the JDL job description. JDL schema does not allow to specify additional parameters to be passed to LRMS. Another problem is that user accounts are assigned dynamically by LCMAPS, therefore there is no guarantee that the users can access reserved resources as they can be mapped to another account. Static mapping is possible, but not recommended. It is possible to create reservations for a Unix group, but then all users from the VO group mapped to these user accounts can access the reservation.

As gLite is strictly bound to Virtual Organisation concepts, it is recommended to use VO to manage users and their privileges with VOMS [7]. VOMS allows for elastic users management and users can have assigned specific roles, groups and attributes. This can be used to reserve some resources for some users by mapping different groups to different resources or can be used to specify a share in the resources by assigning different share and priority to groups. In site configuration, VOMS groups should be statically mapped to UNIX GIDs on CEs, LRMS shares are defined statically according to UNIX GIDs. This approach is simple, but static. It allows VO manager to dynamically assign individual users to different groups in VOMS, but changes in share assignment have to be arranged manually between VO manager(s) and sites. Dynamic share approach was implemented in GPBOX [8], which was a tool that provided the possibility to define, store and propagate fine-grained VO policies. It allowed to map users to service classes and to dynamically change the association between users and classes. Successors of GPBOX are AuthZ and Argus [9]. The working principle is simple. The Argus Authorization Service renders consistent authorization decisions basing on authorization policies defined in XACML. The Policy Administration Point (PAP) provides the tools to author authorization policies. The policies are then automatically propagated to the interested entities, where these are evaluated by a Policy Decision Point (PDP) and enforced by a Policy Enforcement Point (PEP). The VO managers define Group, Roles and capabilities within a VO. Then, they assign users to groups and grant them the possibility to ask for roles (e.g. /vo.plgrid.pl/normalpriority, /vo.plgrid.pl/highpriority). The site administrator on his side defines a set of policies, describing the mapping between service classes (e.g. low, medium, high), local unix groups and LRMS shares. The VO manager defines policy for describing mapping between groups/roles and predefined service classes and this policy can be changed dynamically. XACML semantics allows much more complex policies, not just related to fair share, e.g. for usage quota. By using VO groups, policies and additional VOMS attributes, it is possible to implement the PL-Grid grants concept in gLite.

gLite Implementation of Grid Resource Allocation and Agreement Protocol. The gLite Reservation and allocation architecture was proposed in [10], based on the agreement initiator, agreement service, agreement offer and service provider concepts defined by the Grid Resource Allocation and Agreement Protocol working group of the GGF. The agreement initiator uses the agreement service to obtain appropriate agreements with reservation and allocation service providers,

which are typically co-located with physical or logical resources. In the gLite architecture, agreement initiators would include the workload management system (WMS), the data scheduler (DS), and the user, while reservation and allocation service providers would be associated with the logical representation of physical resources: the computing element (CE), the storage element (SE), and the network element (NE). Attributes defining reservation and co-allocation requests need to be specified in the agreement template/offer, which is initially expressed in JDL. Later on, during the translation to XML, attributes are placed in the Terms section of the agreement offer [WS-AG] and can belong to two alternative subsections depending on their nature: the Service Description Terms (SDTs) and Service Properties subsections. The Terms section provides a quantitative description of the service requested. Both SDTs and Service Properties need to be mapped to corresponding JDL attributes. The agreement template/offer expressed in JDL can specify general attributes: Type ("reservation", "allocation" or "coallocation"), ServiceCategory ("computeElement", "networkElement", "storageElement") and Functionality (e.g. "virtualLeasedLine", "spaceManagement", "bulkTransfer"). For each functionality, a set of specific attributes can be specified, e.g. "DurationTime", "SizeOfTotalSpaceDesiredInBytes", "Bandwidth", "FileTransferEndTime.

The proposed extension was not yet implemented in Workload Management System WMS [11] and computing elements like CREAM.

4.2 Advance Reservation and Co-allocation of Resources Capabilities in QosCosGrid Middleware

The QosCosGrid (QCG) middleware [12,13] 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, multi-scale, MPI or hybrid MPI-OpenMP. However, for many application scenarios the typical best-effort model to access computational resources is not satisfactory, and they require more advanced one, guarantying the requested level of quality of service. Addressing such requirements, QCG, as a first grid middleware offering access to PL-Grid resources, has exposed advance reservation capabilities of the underlying Local Resources Management Systems to end-users.

Researchers can benefit from advance reservations offered by QCG in many ways. Firstly, the advance reservation can be directly used to book in advance resources for a specific period of time. The scenario corresponds to the situation, in which the scientist wants to perform series of experiments in known *a priori* time frame. The time is here usually determined by any event. For example, the access to resources must be synchronized with availability of some equipment, date of presentation or lecture. To the created in such a way reservation one can later submit many tasks to be started without typical delay caused by waiting in a queue.

While requesting for the reservation, a user can specify a list of potential resources (so called candidate hosts) as well as resource and time requirements. QCG can automatically search over all candidate hosts, within user-defined time window, for free resources and for the requested period of time. With QCG it is possible to reserve either a given number of slots located on arbitrary nodes or to request for particular topology by specifying number of nodes and slots per node.

At present, the advance reservations can be created and managed using command-line tools (the QCG-SimpleClient client) or graphical, calendar like, QCG-QoS-Access web application (the Reservation Portal) [14]. Both these tools are clients to the QCG-Broker service and can be used to create new reservations as well as to manage existing ones.

In the QCG-QoS-Access portal, the user can create a new reservation with the intuitive dialog, in which he can specify all requirements and preferences – see Fig. 1. The created reservations are displayed in the portal in a form of a list, where each position includes information about current status of reservation as well as provides a report about reserved resources. If the user wants to resign from the reservation, he can cancel it and release blocked resources.

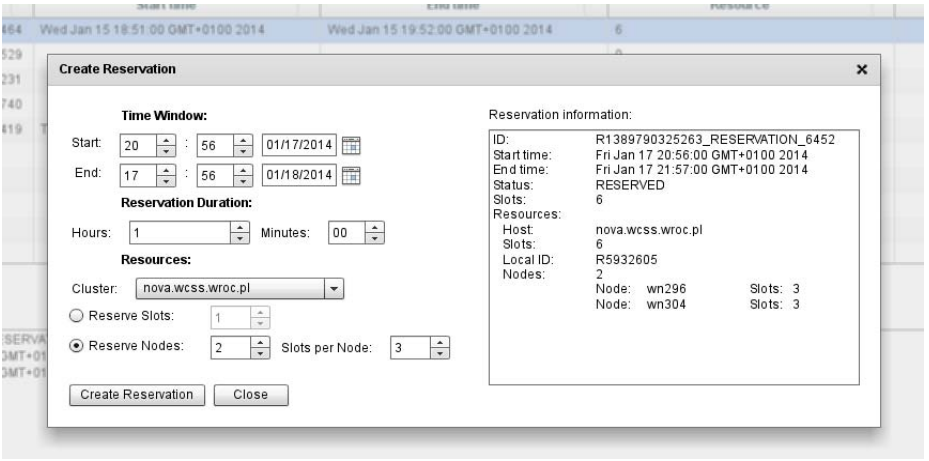

Fig. 1. The QosCosGrid reservation portlet

Creation and management of reservations can be also performed using QCG-SimpleClient. The QCG-SimpleClient is a set of command line tools, inspired by the simplicity of batch system commands. The tools are dedicated to end-users familiar with queuing systems and preferring the command line interface over graphical or web solutions. The learning effort needed to start using QCG-SimpleClient is relatively small as the commands are modeled after the ones known from batch systems. The qcg-* command-line tools allow a user to submit, control and monitor jobs as well as to create and manage reservations. The complete list of commands

can be found in [12]. In the context of the advance reservation, the following tools are particularly important:

- qcg-reserve – creates the reservation and returns its identifier,
- qcg-rcancel – cancels the given reservation(s),
- qcg-rinfo – displays comprehensive information about the given reservation(s),
- qcg-rlist – lists reservation in the system meeting defined criteria.

Every reservation request has to be described in a formal way. The default description format supported by QCG-Client is QCG-Simple. The format does not allow yet to describe more sophisticated scenarios like co-allocation of resources (supported by the XML format called QCG-JobProfile), but is fully sufficient for most of typical cases. The QCG-Simple format description file is a plain BASH script annotated with #QCG directives, what is also a common approach for all modern queuing systems. The #QCG directives inform the system how to process the task and, in case of the reservation, about user's requirements and preferences. Listing 1 presents an example of a QCG-SimpleClient reservation request for 4 slots on *nova* cluster for one hour on 2014.01.25, between 8 am and 4 pm.

```
#QCG host=nova
#QCG walltime=PT1H
#QCG procs=4

#QCG not-before=2014.01.25 8:00
#QCG not-after=2014.01.25 16:00
```

Listing 1. An example of a QCG-Simple reservation description

Detailed information about the created reservation can be obtained with qcg-rinfo. The output of this command for our example reservation is presented in Listing 2.

In contrary to the scenario presented above, in which creation of a reservation and submission of jobs to it are separated steps, QCG also supports the scenario, where the reservation is created especially for a job as a part of submission process. This approach allows to provide the requested level of quality of service with granularity of a single task and to automate the process of managing resources. In such a case, the reservation directives extend directly the task description, while the created reservation is automatically canceled by the system at the end of task execution.

Except direct usage of the advance reservation by end-users, it can be also exploited internally by QCG services for more advanced scenarios like cross-cluster execution of parallel or multi-scale applications. For such applications, distribution across many resources may be required for two reasons. The first one is related to the heterogeneous resource requirements of processes constituting

```
qcg-rinfo R1389951946104__2181

UserDN: /C=PL/O=PL-Grid/O=Uzytkownik/O=PCSS/CN=Tomasz Piontek/CN=plgpiontek
SubmissionTime: Fri Jan 17 10:45:46 CET 2014
DescriptionType: QCG_SIMPLE
StartTime: Sat Jan 25 08:00:00 CET 2014
EndTime: Sat Jan 25 09:01:00 CET 2014
Status: RESERVED
TotalSlotsCount: 4
InUse: false

HostName: nova.wcss.wroc.pl
ProcessesGroupId: qcg
SlotsCount: 4
LocalReservationId: R5949627
Node: wn448 SlotsCount: 2
Node: wn452 SlotsCount: 2
```

Listing 2. An example output of qcg-rinfo command

an application, what is tightly connected with the QCG support for groups of processes and communication topologies. The second one addresses the problem of decomposition of a big task among many resources to enable more complex problem instances, decrease cluster "defragmentation" and to improve resource utilization on the whole system level.

The reservation mechanism is applied in the scheduling process to co-allocate resources and then to synchronize execution of application parts in a multi-cluster environment. QCG supports both the strict and best-effort approaches to resource reservation. In the former approach, resources are reserved only if user's requirements can be fully met (also known as the "all or nothing" approach), whereas in the latter case, the system reserves as much resources as possible, but gives no guarantee that all requested resources (cores) will be available. This feature allows to construct flexible algorithms of processes allocation, in which a whole group of processes can be assigned to a single node or even distributed across many clusters.

QosCosGrid successfully integrates various services and aforementioned tools to deliver to PL-Grid users an e-Infrastructure capable of dealing with various kinds of computationally intensive simulations, including ones that require the requested quality of services. The high-level architecture of the QCG middleware is shown in Fig. 2.

In general, the middleware consists of two logical layers: grid and local one. The basic advance reservation capabilities are offered by the local-level QCG-Computing service, usually deployed on access nodes of batch systems (like Torque or SLURM). The service provides remote access to capabilities of local batch systems. The job submission capabilities of QCG-Computing are exposed via an interface compatible with the OGF HPC Basic Profile [15] specification, while the integration with a queuing system is realized using DRMAA [16]. As

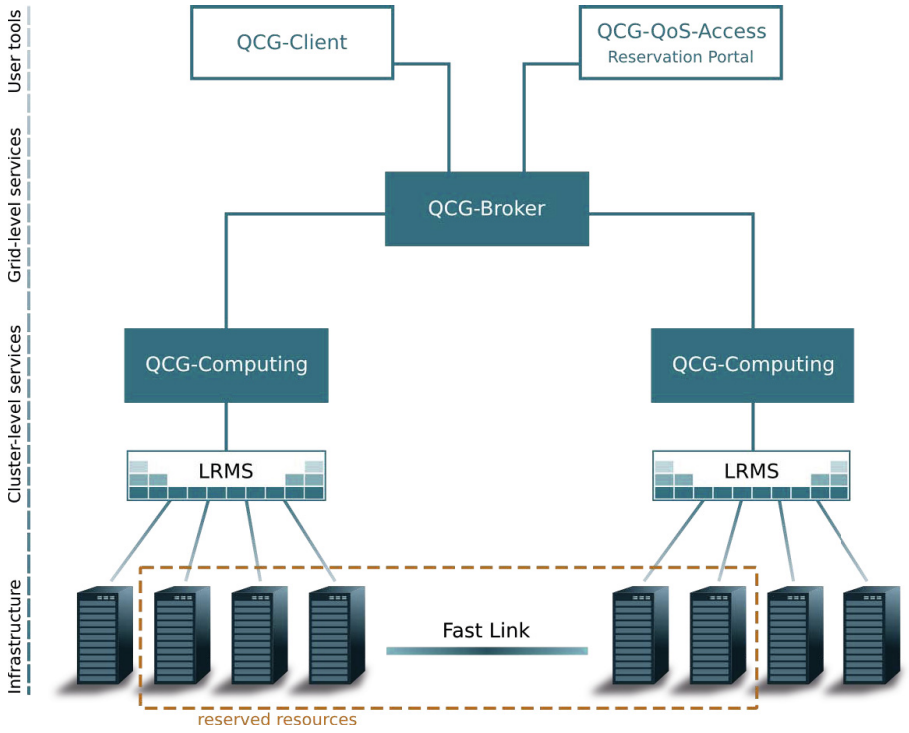

Fig. 2. The QosCosGrid middleware architecture

the first version of DRMAA specification does not address the advance reservation approach, the QCG-Computing specific interface (with a dedicated description language) was proposed to support this capability. Currently, in QCG, advance reservations are created by calling LRMS scheduler commands directly, while in the future leverage of Advance Reservation API of Open Grid Forum DRMAA 2.0 [17] specification is planned. What is important, flexible configuration allows the local system administrators to keep the full control over resources that can be reserved, limiting for example advance reservation capabilities only to a single system partition.

More advance scenarios, like reservations in multi-cluster environment and co-allocation of resources, are supported by the grid-level service, called QCG-Broker, which benefits from QCG-Computing capabilities for a single cluster. The QCG-Broker service, using the adaptive mechanism to determine a time window for a reservation, tries to allocate resources on machines meeting user's requirements. In order to gather the requested amount of resources, the procedure of selection is performed in a loop. If the amount of reserved resources is not satisfactory, the resources are released and the whole procedure is repeated for the next time window.

Within the MAPPER project [18], the QosCosGrid stack has been integrated with the MUSCLE library [19], enabling cross-cluster execution of so-called

multi-scale applications. The common multi-scale application consists of number of single-scale modules that calculate some phenomena on different spatial or temporal scales and simultaneously exchange information with each other. 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.

4.3 Reservations in UNICORE

The UNICORE [20] server side included basic resource reservation support for a long time, however only the Maui scheduler was supported and the reservation functionality was not enabled out of the box, conversely, it required manual integration. Especially, the client side support was missing, making the feature unusable without a dedicated development effort.

Since the version 6.5.1 of the UNICORE servers release (around the end of 2012), the resource reservations support was enhanced to support SLURM and is integrated by default in the official UNICORE distribution. This work was greatly influenced by the input and contributions coming from the PLGrid Plus project. At the same time, the resource reservation control interface was added to the UNICORE Command line Client (UCC).

The infrastructure is shown in Fig. 3.

UNICORE support for resource reservations is divided into two distinct parts: reservations management support and submission of jobs to a reservation. This

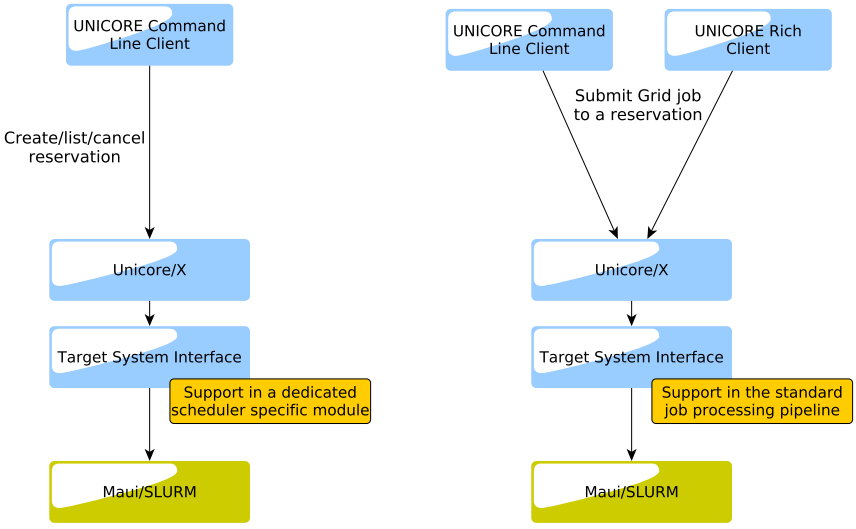

Fig. 3. UNICORE resource reservation processing: management (left side) and usage (right side)

approach covers a common situation where resource reservation functionality is not directly exposed to end-users and they can only submit jobs to reservations created by the site staff or external tools.

The UNICORE server side advertises for each grid site whether it supports reservations or not. For those sites supporting reservations, a user can create a reservation, list owned reservations and delete some of the previously created. It is worth to underline here that the complex reservation resources as CPUs, nodes or duration are all specified using the same syntax as the resource requirements for an ordinary UNICORE job. Therefore, the user is not faced with the differences between various schedulers.

Submitting a job to a reservation is a simple task: it is enough to set the reservation identifier in the job's resource requirements. While the reservations management support is only available in the UCC client, the submissions of jobs to reservations is available in both UCC and UNICORE Rich Client (URC).

Server side reservations handling is performed in a similar way to job processing: the Unicore/X Web Service component is the site's entry point talking to clients. It forwards the requests to the Target System Interface (TSI) server, which maps an abstract grid reservation related operation to something meaningful to the scheduler being used. It should be noted here that the reservation related operation, while similarly handled as a classic job, has its own (simplified) processing pipeline. This approach is probably correct, taking into account the fundamental lifecycle differences between a resource reservation and a grid job, however it also results in some limitations. Probably the most important one is that the functionality of the UNICORE incarnation tweaker[1] subsystem is not available for reservations.

The most significant contribution of the PLGrid Plus project to UNICORE resource reservations subsystem was a complete UNICORE TSI reservations management module for SLURM. What is more, the Maui module was updated and fixed in several places.

The most difficult parts of reservations handling in UNICORE are related to reservations accounting and authorization of reservation management. In the PL-Grid Infrastructure we have decided not to include any of those functions in the grid layer. This decision was dictated by the fact that direct access to computing sites is generally possible, so the accounting and authorization must be anyway solved on the lower, resource scheduler layer. Still, UNICORE provides some integration points with respect to those issues. It is possible to authorize reservation management operations on the Web Service level using the standard UNICORE authorization policy. The limitation of this approach is that the authorization is coarse grained: only the complete functionality access can be controlled, it is not possible to authorize basing on particular reservations or parameters (e.g. to ban reservations longer then a given value).

[1] Incarnation tweaker is a UNICORE server's feature allowing for nearly unlimited inspection and modification of the submitted job. It is used to fix common mistakes in a job description, add site specific settings, enforce required options and finally trigger additional actions for selected jobs.

The accounting of reservations in UNICORE is not supported. The only integration point is the TSI script, which can be modified to invoke accounting operations. However, besides the statistical knowledge about the amount of reservations made through UNICORE, the actual accounting of resource reservations should be made on the scheduler level to accommodate all reservation changes made externally to UNICORE.

We can conclude with the statement that our evaluation and deployment of UNICORE reservations was successful. All the basic features are currently enabled in the infrastructure and we support both Maui and SLURM schedulers, which are deployed in PL-Grid. The generally available SLURM support in UNICORE was contributed by the PLGrid Plus project. Beside those achievements, we can also point out limitations of the solution: PBS Pro is not supported while it is used by one of the PL-Grid Infrastructure sites. There is no support for higher, grid-level reservations. Functionality to broker reservations (i.e., to create them at any site fulfilling the given reservation resource requirements) is missing and could improve the user experience as well as the support for coordinated multi-site reservations. Nevertheless, we can underline that the above problems are rather minor, taking into account that not all sites in the infrastructure allow for the end user controlled resource reservation creation, due to well-known risks (related for instance to computational resource wasting).

5 Summary and Future Work

The functionality of the advance reservation is very comfortable for the user. However, it should be used in an efficient way in order not to waste resources at HPC clusters. Thus, usage of reserved resources is monitored and accounted. The user negotiates with a resource provider separately wall clock time to be spent on reserved resources and total time in the PL-Grid grant. The time spent in batch jobs is accounted based on PBS standard logs. The accounting of reservations requires additional logging. In Moab, they are triggered by pre-agreed actions, e.g. reservation ready, reservation removed. The next step would be integration of these two sources of information about reservations and jobs (PBS logs and reservation logs) to avoid double charging the user for jobs executed within reservation and to charge the user for unused reservations.

References

1. Foster, I., Kesselman, C., Lee, C., Lindell, B., Nahrstedt, K., Roy, A.: A Distributed Resource Management Architecture that Supports Advance Reservations and Co-allocation. In: Proceedings of the 7th International Workshop on Quality of Service, London, UK (1999)
2. XSEDE project web page, `https://www.xsede.org/overview`
3. Metascheduling Requirements Analysis Team report, `http://www.teragridforum.org/mediawiki/images/b/b4/MetaschedRatReport.pdf`

4. MacLaren, J.: HARC: The Highly-Available Resource Co-allocator. In: Meersman, R. (ed.) OTM 2007, Part II. LNCS, vol. 4804, pp. 1385–1402. Springer, Heidelberg (2007)
5. Casajus, A., Graciani, R., Paterson, S., Tsaregorodtsev, A.: DIRAC Pilot Framework and the DIRAC Workload Management System. Journal of Physics, Conference Series 219, 062049 (2010)
6. The Grid Resource Allocation and Agreement Protocol Working Group, Global Grid Forum, https://forge.gridforum.org/projects/graap-wg
7. VOMS home page, http://italiangrid.github.io/voms/
8. Guarise, A.: Policy management and fair share in gLite. In: HPDC 2006, Paris (2006)
9. Argus Authorization Service, https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework
10. Ferrari, T., Ronchieri, E.: gLite Allocation and Reservation Architecture. EGEE JRA1 technical report (2005), http://edms.cern.ch/document/508055
11. Job Description Language Atrribute Specification for the Workload Management System, https://edms.cern.ch/file/590869/1/WMS-JDL.pdf
12. Bosak, B., Kopta, P., Kurowski, K., Piontek, T., Mamoński, M.: New QosCosGrid Middleware Capabilities and Its Integration with European e-Infrastructure. In: Bubak, M., Kitowski, J., Wiatr, K. (eds.) PLGrid Plus. LNCS, vol. 8500, pp. 34–53. Springer, Heidelberg (2014)
13. Bosak, B., Komasa, J., Kopta, P., Kurowski, K., Mamoński, 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, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 40–55. Springer, Heidelberg (2012)
14. Kurowski, K., Dziubecki, P., Grabowski, P., Krysiński, M., Piontek, T., Szejnfeld, D.: Easy Development and Integration of Science Gateways with Vine Toolkit. In: Bubak, M., Kitowski, J., Wiatr, K. (eds.) PLGrid Plus. LNCS, vol. 8500, pp. 147–163. Springer, Heidelberg (2014)
15. HPC Basic Profile Version 1.0, http://www.ogf.org/documents/GFD.114.pdf
16. 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, CCGRID 2007, pp. 619–626. IEEE Computer Society, Washington, DC (2007)
17. Distributed Resource Management Application API Version 2 (DRMAA), http://www.ogf.org/documents/GFD.194.pdf
18. Ben Belgacem, M., Chopard, B., Borgdorff, J., Mamoński, M., Rycerz, K., Harężlak, 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)
19. Borgdorff, J., Mamoński, M., Bosak, B., Kurowski, K., Ben Belgacem, M., 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)
20. Streit, A., et al.: UNICORE 6 – Recent and Future Advancements. Annals of Telecommunications 65(11-12), 757–762 (2010)