LCG AND ARC MIDDLEWARE INTEROPERABILITY

M. Grønager*, UNI-C, Lyngby, Denmark
A. Wäänänen, Univ. of Copenhagen, Denmark
T. Ekelöf, M. Ellert, Uppsala Univ., Uppsala, Sweden
B. Kónya, O. Smirnova, Lund Univ., Lund, Sweden
F. Ould-Saada, S. Haug, A. Konstantinov, Univ. of Oslo, Oslo, Norway
L. Field, D. Smith, CERN, Geneva, Switzerland
R. Walker, Simon Fraser Univ., Vancouver, Canada

Abstract

LCG and ARC are two of the major production-ready Grid middleware solutions being used by hundreds of HEP researchers every day. Even though the middlewares are based on same technology, there are substantial architectural and implementational divergences. An ordinary user faces difficulties trying to cross the boundaries of the two systems. LCG clients so far have not been capable accessing ARC resources and vice versa. After presenting the similarities and differences of the LCG and ARC middlewares, this paper focuses on the strategies for implementing interoperable layers between the two solutions. The most important areas are job submission and the information system. The basic requirement for the interoperability layer is the capability for transparent job submission from LCG to ARC.

INTRODUCTION

The ARC[1] and LCG[2] middleware projects shared a common background in the early stages of the European Data Grid project[3]. LCG is an evolution and ARC is an actual spawn. The original aim, however, remained the same. To primarily enable the data acquisition and processing of the four experiments connected to the Large Hadron Collider at CERN and by achiving this goal, also enable the use of the middleware by other research communities.

The LCG middleware is a deployment ready software stack built from components of other middlewares. The most important are those from EDG, followed by elements of AliEn[4] and the middleware developed by the EGEE[5] project, gLite[6]. In addition, LCG in its various versions often offers several different solutions for the same middleware component, simplifying transition from one implementation to another. The version numbering scheme of LCG continues that of EDG, and around summer 2006 LCG will change name to gLite version 3, emphasizing the introduction of more and more services from gLite, though still leaning on the EDG heritage. In this paper, the comparison of the middlewares is based on LCG version 2.6, but the proposed scheme for interoperability is based on components to be introduced in gLite 3.

The ARC middleware was born from the Nordic Council of Ministers' supported project "Nordic Testbed for Wide Area Computing and Data Handling" which ran in 2001-2003 and gave rise to the NorduGrid collaboration[1]. It aimed to ensure Nordic participation in the EDG project. With the help of the Nordic Data Grid Facility[7], the ARC middleware was deployed on the majority of the Nordic computing facilities and in 2002 was the first and largest production Grid in the world. Today there are several resources using ARC outside the Nordic countries as well. Most compute resources in the Nordic countries are using the ARC middleware, although a few resources recently also became accessible also through LCG middleware.

In this study, version 0.4.5 of ARC is the basis for the comparison.

One of the LHC experiments, the ATLAS experiment, has conducted several production test runs of its application-specific software. One of these, ATLAS Data Challenge 2, used resources in US and Europe and as such had to use several middlewares. For this purpose a meta-scheduler, the ATLAS Production System[8], was created, enabling job-submission to the three grids: Grid3 (now OSG[9]), LCG and NorduGrid.

It is the desire to streamline this work into actual interoperability between several grids that motivates the OSG LCG interoperability as well as this activity.

The first step is to compare the two grid middlewares and understand the differences. After this, Second, is to work out how these differences can be bridged. The overall goal from this is to understand what standards are required to retain interoperability.

MIDDLEWARE COMPARISON

LCG and ARC shared a common background in the European Data Grid project and hence also share several of the basic building blocks.

The most basic building block of ARC 0.4.5 and LCG 2.6.0 is the Globus Toolkit version 2.4 (GT2) [10]. Globus by itself is not a deployment ready grid solution, rather it is a toolkit for building other grid middleware. GT2 defines a security scheme (GSI – Globus Security Infrastructure)[10] for user authentication, authorization and delegation of rights. The scheme is based on X509 certificates and short lived proxy certificates[11] inspired by the Kerberos 5[12] security framework.

GT2 also defines protocols for file-transfer (GSIFTP), job-submission (GRAM) and an information system based on LDAP(MDS) [10]. Finally, GT defines a resource specification language, RSL.

ARC and LCG use the same security framework based on the GT2 GSI and support the Virtual Organisation extensions to GSI, VOMS[13], which was created during the EDG project. The file transfer protocol gsiftp is also used by both middlewares, and enables the transfer of files between storage elements of LCG and ARC.

*gronager@nbi.dk

Consider table 1 in which the different protocols, schemas and descriptions of LCG and ARC are listed. Except for security and file transfer protocols, ARC and LCG differ in the other areas, , although in some areas the differences are less important. When GT2 componets are proven not to handle a production scenario, a different solution was introduced by each project to replace the GT2 component with something that was production quality. This resulted in divergence between the different middlewares.

The job submission component in GT2 did not scale. The gateway spawns a process for each job running on a cluster, which for bigger clusters might result in hundreds of processes running, rendering the gateway completely unresponsive, further file staging capabilities were very limited in the first versions. ARC solves these problems by introducing its own file-staging capalble grid-manager and by using gsiftp, with a custom plugin to submit jobs.

The same problem was tackled within LCG by using Condor-G, from Condor[14], as a submission protocol. Condor-G uses a series of Globus commands to kill the processes spawned per job on the gateway and replaces this with a process per user, which in most cases scales much better as for production scenarios as one user typically submits several hundreds jobs[8].

Both LCG and ARC rely heavily on the information system as both push jobs to the resources. The original GT2 MDS (Monitoring and Discovery System) is based on several LDAP servers. Those located at sites and containing resource information (GRIS) and indexing servers which aggregate the information (GIIS). They are usually organized in a tree-like cached hierarchy enabling complete information retrieval by querying the top level GIIS node. However, when multiple clients request information, GIIS-es become seriously overloaded due to serving multiple recursive LDAP searches. Further internal queries for updating the cache tends to timeout and cause stability problems.

ARC and LCG have solved this problem in different ways. ARC uses a top level GIIS only to index the actual sites which are then queried in parallel by the brokering client, hence no caching is involved. LCG has replaced the hierarchical caching by a single cache database the BDII[15]. Both schemes solve the lack of scalability in the GT2 approach.

GT2 offers templates for representing only compute host resources in the information system. However, a more complete information schema was needed . ARC solved this by a using hierarchical schema describing clusters, queues, jobs and users. This schema is usually referred to as the nordugrid-schema[16]. The GLUE schema[17] that is used by LCG originates from the work performed within the EU DataTAG project, and is substantially different from that of ARC.

In the area of brokering, GT2 offers no solution, thus every project has to come with own implementations. One of the most notable differences between ARC and LCG is the completely different approach to brokering. ARC brokering is done solely by the user interface clients, where LCG has a resource broker service for a large number of user interfaces. To understand the motivation behind these two choices, it is worthwhile considering the usual requirement for a grid. The user should be able to submit a job from a user interface – e.g. a laptop, go offline and then later check the job progress, probably from a different location. This means that no connection can be held open between the resource and the user interface. This cannot be fulfilled by GT2 since server initiated back connections will be made. The same limitations also apply to the GRAM via Condor-G submission. Hence, there is a need to either introduce a submission server, which also can do the brokering, for example the LCG Resource Broker, or to use a different protocol other than GRAM, like the ARC gsiftp job submission.

The introduction of a special server for resource brokering adds an extra bottleneck, however, it also enables much more detailed and automated job monitoring and inter-job dependencies as can be described with DAG jobs[18].

For job description, GT2 uses RSL (Resource Specification Language), which mostly describes the resources needed by a job. ARC has adapted this to become a job description language, adding extra attributes to enable specification of e.g. runtime environments etc. LCG followed EDG in basing matchmaking and brokering on Condor ClassAds[19], which enables a fine description of jobs and resources. Hence it was straightforward to choose the Condor job description language (JDL) to describe jobs and resource requests. Before the actual submission to the Globus-based compute elements it is, however, translated into Globus-RSL.

Functionality	**ARC**	**LCG**
Security	GSI+VOMS	GSI+VOMS
Info-system	LDAP+ARCGIIS	LDAP+BDII
Info-scheme	ARC	GLUE 1.2
Job-submission	GSIFTP	GRAM/Condor-G
Job-description	xRSL	JDL
Brokering	by client	by RB
File transfer	GSIFTP	GSIFTP
Resource Manager	ARC Grid Manager	Globus Job Manager

Table 1: Comparison between ARC and LCG. The Resource Manager is the service at the local resource responsible for managing jobs on the resource.

ENABLING LCG TO ARC JOB SUBMISSION

In order to enable brokering and submission to ARC computing resourcess (CEs), several issues need consideration. In Table 1. one can easily see the differences that need to be addressed. The first target naturally falls on how to submit jobs – via the client or via the resource broker. To best follow the LCG scheme we chose to submit via the resource broker, and since a change from the old LCG-RB to the new gLite-RB was already planned, the gLite-RB should be used.

Submission from the resource broker to the ARC CEs first requires that the ARC information system is imported into the BDII used by the resource broker. This was accomplished by doing a small modification to the BDII that translates the ARC schema to the Glue Schema between querying the site and adding the information to the BDIIs database. The translation key is listed in Table 2. After translation the result is made availabe in the BDII.

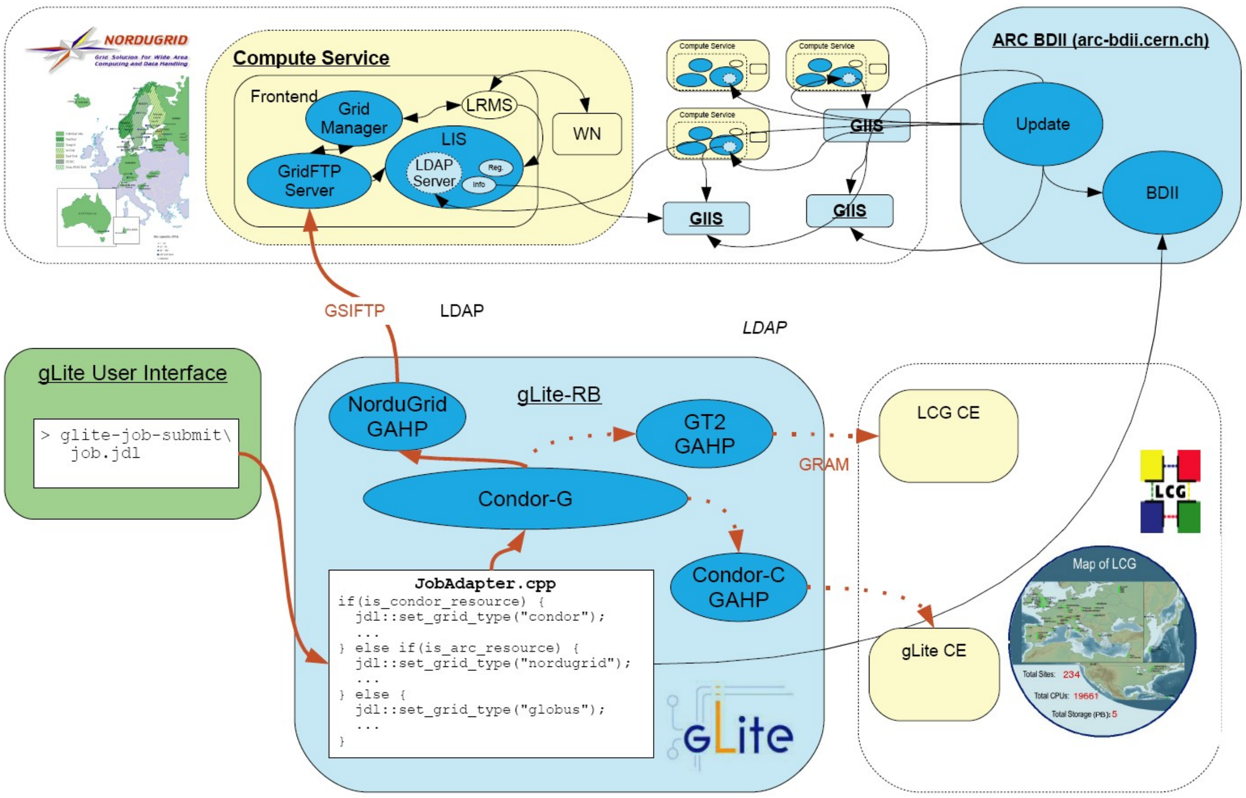
A similar approach was used for cross grid brokering between LCG and ARC previously [20]. Instead of translation into GLUE, the information was fed directly into a Condor collector process.

GLUE	ARC
GlueSiteUniqueID	nordugrid_cluster_location
GlueSiteName	nordugrid_cluster_location
GlueSiteDescription	nordugrid_cluster_location
GlueSiteUserSupportContact	nordugrid_cluster_support
GlueSiteSysAdminContact	nordugrid_cluster_support
GlueSiteSecurityContact	nordugrid_cluster_support
GlueClusterUniqueID	nordugrid_cluster_name
GlueClusterName	nordugrid_cluster_name
GlueClusterService	nordugrid_cluster_name
GlueSubClusterUniqueID	nordugrid_cluster_contactstring
GlueHostApplicationS.w.R.T.Env.	nordugrid_cluster_runtimeenv.
GlueCEUniqueID	nordugrid_cluster_contactstring
GlueCEHostingCluster	nordugrid_cluster_name
GlueCEName	nordugrid_cluster_aliasname
GlueCEInfoHostName	nordugrid_cluster_name
GlueCEInfoLRMSType	nordugrid_cluster_lrms_type
GlueCEInfoLRMSVersion	nordugrid_cluster_lrms_version
GlueCEInfoTotalCPUs	nordugrid_cluster_totalcpus
GlueCEStateRunningJobs	nordugrid_cluster_usedcpus
GlueCEStateTotalJobs	nordugrid_cluster_totaljobs
GlueCEStateWaitingJobs	nordugrid_cluster_queuedjobs
GlueCEPolicyAssignedJobSlots	nordugrid_cluster_usedcpus

Table 2. The Translation key between GLUE and ARC schemas. This table is only the most important minimal subset, which was used in the first translation scripts. For the full GLUE to ARC translation key, please consult [16]

The information retrieval process takes around 2 seconds and the translation process is 6 seconds. The process is repeated every 2 minutes.

The gLite Resource Broker can query information about ARC CEs in a similar manner as for the LCG CEs and hence the ARC CEs are also used in the brokering. If an ARC CE is chosen by the brokering algorithm, the job will be adapted to submission via Condor. The adaptation is done for submission to all CE types, whether an GT2 based LCG CE, a Condor-BLAHP based gLite CE or as in the present case, submission via gsiftp to an ARC CE. Condor manages the actual submission and adaptation. Thus the only thing needed is to inform Condor of the "grid type" for the submission target. Figure 1 illustrates this and outlines the important part of the file JobAdapter.cpp in where this selection is made. Currently, Condor (version 6.7.15) supports GT2, GT3, GT4, NORDUGRID and CONDOR-C submission targets. Given the chosen target is NORDUGRID, Condor submits the job via Condor-G, transforms the JDL into xRSL and does the actual submission via the gsiftp protocol.



```
> glite-job-submit\
   job.jdl
```

```
JobAdapter.cpp
if(is_condor_resource) {
  jdl::set_grid_type("condor");
  ...
} else if(is_arc_resource) {
  jdl::set_grid_type("nordugrid");
  ...
} else {
  jdl::set_grid_type("globus");
  ...
}
```

Figure 1: The proposed LCG->ARC interoperability setup. Jobs are submitted through gLite to the gLite resource broker. The BDII used for the brokering has been updated with information obtained and translated form ARC compute elements. With this brokering information a proper site is chosen. If the site is of type "nordugrid" a condor classadd "grid_type=nordugrid" is inserted resulting in submission through gsiftp.

Figure 1 shows the complete LCG to ARC interoperability chain: A user submits a job using `glite-job-submit`. The job then enters the gLite resource broker where the job requirements are compared to the advertised classads in the BDII. The BDII has been updated with data from the ARC information system. If an ARC CE is found to best suit the job requirements the job enters the JobAdapter. From here all the job specifications and BDII information are filled into Condor classad structures and in particular, the grid_type is set to "nordugrid". The job now enters Condor-G, which upon resolving ClassAds uses the NorduGrid-GAHP (Grid ASCII Helper Protocol) to submit the job to the ARC CE using gsiftp job submission. While the job is running, the NorduGrid-GAHP adapter runs on the Resource Broker and monitors job state. On completion, the job output is transferred back to the resource broker, and becomes available for the user interface.

STANDARDS AND CONCLUSIONS

We have described the differences and similarities between LCG and ARC grid middlewares and explained the course for this in a historical view.

Furthermore, an interoperability setup for doing LCG to ARC cross grid job submissions was presented. The scheme aims for a high degree of interoperability where an ARC CEs can be used as a submission target on the same level as a classic LCG CE or the new gLite CE.

The aim for this interoperability work has been to enable cross grid submission on a short time scale. In the long term , the efforts of the Global Grid Forum initiatives in general and especially the MultiGrid working groups on Interoperability should work towards common interfaces. Further, on the longer term we plan to add ARC to LCG job subission and further work on job-management.

Logging and bookkeeping has not been addressed in this paper, as we expect that the work towards standards in this area will lead to interoperability between LCG and ARC without the need for special adapters. Furthermore, we expect that the current efforts on the GLUE2 schema will eliminate the need for the information system translation step as presented here.

ACKNOWLEDGEMENTS

This work was supported by the EGEE, EGEE-II and KnowARC EU projects. Further, the authors which to thank E. Laure, M. Schulz and I. Bird, CERN for fruitful discussions and input within the ARC-LCG interoperability task force.

REFERENCES

[1] http://www.nordugrid.org/.
[2] http://lcg.web.cern.ch/LCG/.
[3] http://eu-datagrid.web.cern.ch/eu-datagrid/
[4] http://alien.cern.ch/
[5] http://public.eu-egee.org/
[6] http://glite.web.cern.ch/
[7] http://www.ndgf.org/
[8] R. Sturrock et al., "Performance of the NorduGrid ARC and the Dulcinea Executor in ATLAS Data Challenge 2", in A.~Aimar and J.~Harvey and N.~Knoors, Proc. of CHEP 2004, CERN-2005-002, 2005, p. 765-768
[9] http://www.opensciencegrid.org/
[10] http://www.globus.org/
[11] RFC3820
[12] http://web.mit.edu/kerberos/
[13] http://grid-auth.infn.it/docs/VOMS-Santiago.pdf
[14] http://www.cs.wisc.edu/condor/
[15] L. Field, M. W. Schulz, “Grid Deployment Experiences: The path to a production quality LDAP based grid information system”, in A.~Aimar and J.~Harvey and N.~Knoors, Proc. of CHEP 2004, CERN-2005-002, 2005
[16] http://www.nordugrid.org/documents/arc_infosys.pdf
[17] http://infnforge.cnaf.infn.it/glueinfomodel/index.php/-Spec/V12
[18] http://www.cs.wisc.edu/condor/dagman/
[19] R. Raman et al., “Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching”, Proc. of the Twelfth IEEE Int. Sym. on High-Perf. Dist. Comp., June, 2003, Seattle, WA
[20] R. Walker et al., “A Grid of Grids using Condor-G”, Proc. of CHEP 2006, (to be published)