Job invocation interoperability between NAREGI Middleware Beta and gLite

Hidemoto Nakada
National Institute of Advanced Industrial Science and Technology (AIST)
1-1-1 Umezono, Tsukuba, 305-8568, Japan
hide-nakada@aist.go.jp

Hitoshi Sato
Tokyo Institute of Technology,
2-12-1 Ookayama, Tokyo, 152-8550, Japan
hitoshi.sato@is.titech.ac.jp

Kazushige Saga
2-1-2 Hitotsubashi, Tokyo 101-8430, Japan
saga@grid.nii.ac.jp

Masayuki Hatanaka
140 Miyamoto, Numazu 410-0396, Japan
hatanaka@soft.fujitsu.com

Yuji Saeki
2-1-2 Hitotsubashi, Tokyo 101-8430, Japan
ysaeki@grid.nii.ac.jp

Satoshi Matsuoka
Tokyo Institute of Technology,
2-12-1 Ookayama, Tokyo, 152-8550, Japan
matsu@is.titech.ac.jp

Abstract

As grid middleware stacks mature, the importance of inter-operation among them is getting more significant. There is a community group called GIN (Grid Interoperation Now) in the OGF (Open Grid Forum), a standardization body for grid related technologies, which aims to establish interoperation among several grid middlewares. We performed experiments on inter-operation between our NAREGI Middleware beta and EGEE gLite, as a contribution to the group. For the experiments, we implemented several modules to allow information exchange and mutual job submissions. As the results of the experiment, we confirmed the followings: 1) The security layer, such as certificates and virtual organization management, imposes no fundamental difficulties despite the subtle diffrences in the use of proxy certificates, 2) While information services differs substantially, the resource information can be translated to allow effective information exchange via schema translation between GLUE and CIM, 3) Jobs can be mutually submitted based on the exchanged information, despite the differences in job description languages and interfaces (JSDL vs. JDL).

1. Introduction

The concept of grid, connecting distributed resources for higher utilization, is getting widely understood and accepted. There are a variety of grid middleware stacks proposed, and used to actually manage grids. However, most of them can not interoperate with each other, preventing us from fully leveraging the concept of 'grid'. There are several possible causes that hinder interoperation amongst the diverse grid middleware stacks including: 1) differences in the security layer, 2) differences in the resource information schema and notation format, 3) differences in the commnication protocols. For further growth and dissemination of the grid technology it is crucial to establish interoperation technology among various grid middleware stacks.

A community group in the OGF (Open Grid Forum, a standardization body for grid related technologies) [10], called GIN(Grid Interoperation Now) [11], is formed aiming to establish interoperation technologies among several grid middleware stacks and feedback the experience to the standardization groups.

We show the details of an experiment on interoperation between NAREGI Middleware Beta and EGEE [4] gLite [6, 5], which was performed as a contributory research and experiment to the community group. There, we accomplished job submission interoperability from NAREGI Middleware Beta to gLite and from gLite to NAREGI Middleware Beta. We enabled the former by (effectively) turning gLite Computing Elements into NAREGI Middleware Beta resources so that they can be allocated by the NAREGI Middleware Beta brokering system. We enabled the latter by exposing whole NAREGI Middleware Beta as one of the Computing Elements of gLite.

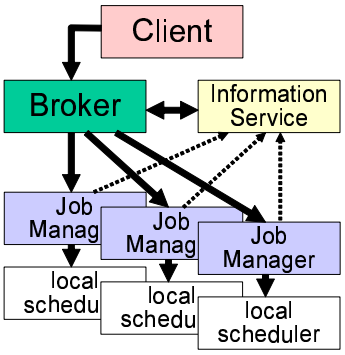

Figure 1. Generic configuration of Grid middlewares

We confirmed the followings through the experiments: 1) the two middleware can interoperate each other in terms of security infrastructure, such as certificates and virtual organization management, despite the minor differences in how proxy certificates are used, 2) information services of them can be bridged with intermediate modules with effective schema translation between GLUE schema employed by gLite and CIM-based schema employed by NAREGI, and 3) jobs can be submitted from one to another via bridging modules, despite the differences in job description languages and interfaces (JDL vs. JSDL).

2. Generic Grid Middleware Configuration and Interoperation

Figure 1 shows the generic configuration of a grid middleware stack: it consists of *brokers*, *job managers* and *information services*. Job managers reside on every site that provide compute resources and act as a bridge to the backend local schedulers. Information service gathers various information, such as load average or queue length, from each site and provides them to the broker, based on a certain information schema. The broker receives job submission requests from users, describes in term of a job description language, and finds compute resources suitable for executing the job based on the information provided by the information service, and forward the job submission request to the job managers running on the compute resources. The job managers pass the jobs to the backend local schedulers so that they are executed. All the communications among the modules has to be properly authenticated and authorized.

Overall, to enable inter-grid middleware job submission, we need to accomplish the followings: 1) security infrastructure interoperation, 2) information service interoperation, and 3) job submission interoperation.

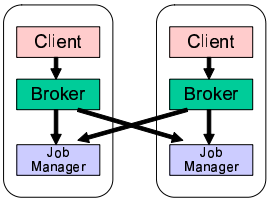

Figure 2. Job Manager Level Interoperation.

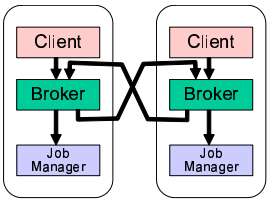

Figure 3. Broker Level Interoperation.

For job submission interoperation, there can be two basic strategies; job manager level interoperation and broker level interoperation. With the former strategy, the broker of the requester middleware stack will submit jobs to the job manager of requestee middleware as shown in figure 2. With this strategy, since the broker of the requester middleware stack determines which compute resource to execute the job, the broker of the requestee middleware loses control over which resource should be used.

With the latter strategy, the broker of requester middleware will submit jobs to the broker of the requestee middleware as shown in figure 3. The jobs will go through both of the brokers, leading to better manageability at the expense of more complexity and possible performance overhead of dual brokerings.

3. NAREGI Middleware Stack Overview

NAREGI Middleware Beta is a grid middleware stack that is designed focusing on co-allocation of resources from multiple-sites and execution of complex workflow of jobs. While the first incarnation of the middleware, which is called *alpha* [17] was based on UNICORE [13], the *beta*, the second incarnation, is based on WSRF [14] and Globus Toolkit 4 [16]. The most notable characteristic is its early adoption of emerging standards mainly from OGF.

The execution resource management portion of NAREGI Middleware Beta is composed of several modules including the followings three primary modules. (Figure 4)

- **Super Scheduler (SS)**
  The Super Scheduler is responsible not only for brokering jobs but also resource co-allocation and workflow execution. Each Virtual Organization in the grid will manage one or more SS(es), and its member users create workflows and submit them to the SS. SS interprets the workflows, finds jobs that can be executed at the point, finds the appropriate resources communicating with IS (Information Service) described below, makes reservations for the jobs communicating with the GridVM below on the found resource, and submit the jobs to the GridVM. If co-allocation spanning over several sites is needed, the SS will automatically find commonly available timeslot for all the sites and makes reservation on the sites.
- **GridVM**[1]
  GridVM is a module that is responsible for management of computation resources in a site, corresponding to the "Job Manager" described in the previous section. It wraps around a local batch queuing system and Globas Toolkit 4 GRAM, supporting the JSDL (Job Submission Description Language) [9] defined by OGF, adding advance reservation interface for the Super Scheduler. It is also responsible for resource consumption control in computer resources, and collecting and notifying resource usage information for accounting.
- **Information Service (IS)**
  NAREGI IS (Information Service) is a framework to aggregate and provide resource information, based on WSRF. All the resource information are represented in an extension of the CIM [1] schema, which is defined by DMTF [3], which is a standardization body for resource representation. The information is stored in the backend relational database. IS provides Information retrieval interface based on OGSA-DAI [12], which is a standardized database access protocol from OGF, and information update protocol based on WSRF. A module called LRPS (Local Resource Provider Service) is responsible for updating the information of each resource.

4. gLite Overview

GLite is a grid middleware that is developed by the EGEE (Enabling Grids for E-Science in Europe) project, and widely deployed amongst almost 50 European and other countries. GLite is composed of several modules including the followings:

1 GridVM is a very thin layer of abstraction for job submission and execution, and is not a full-fledged virtual machine like Xen where OS environments are virtualized

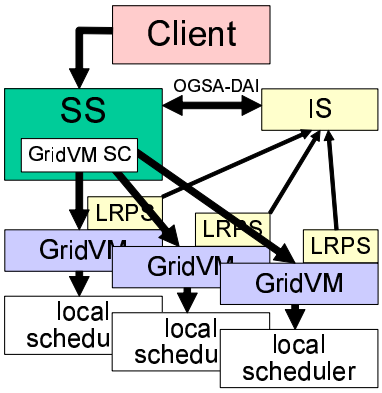

Figure 4. NAREGI Middleware beta overview.

- **WMS (Workload Management System)**
  WMS is a brokering module for gLite. It receives job submission requests from users and queries resource status to BDII, described below, and based on the resource information it allocates resources to jobs. For brokering, it uses ClassAd [20], based matchmaking that is originally proposed and introduced by Condor Project [2].
- **BDII (Berkeley Directory Information Index)**
  BDII is the information service module for gLite, which provides information access interface based on LDAP (Light-weight Directory Access Protocol) [21]. As the data schema for the stored resource information, it employs the GLUE schema [8] , which is a simpler, higher-level abstraction compared to CIM, being more specific to grid properties rather than being detailed and comprehensive as is with CIM
- **CE (Computing Element)**
  This is the job manager module for gLite. As a job description language it supports its own JDL (Job Description Language) format, different from the JDSL standard.

  There are two incarnations of CE, namely LCG-CE and gLite-CE, which have totally different architecture. The LCG-CE is a carried-over module from the sister project to EGEE called LCG (LHC Computing Grids), which is supported for backward compatibility purpose. Effectively, it is a pre-WS GRAM [15] of the Globus Toolkit. The gLite-CE, which is a new tailored module for gLite, is a complex module that uses Globus pre-ws GRAM protocol and the Condor submission protocol at the same time.

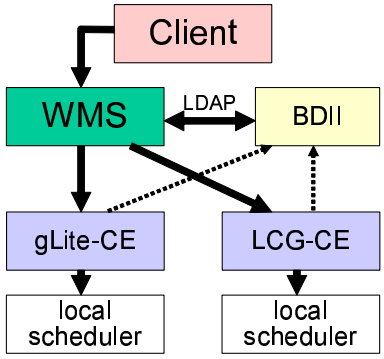

Figure 5. gLite overview.

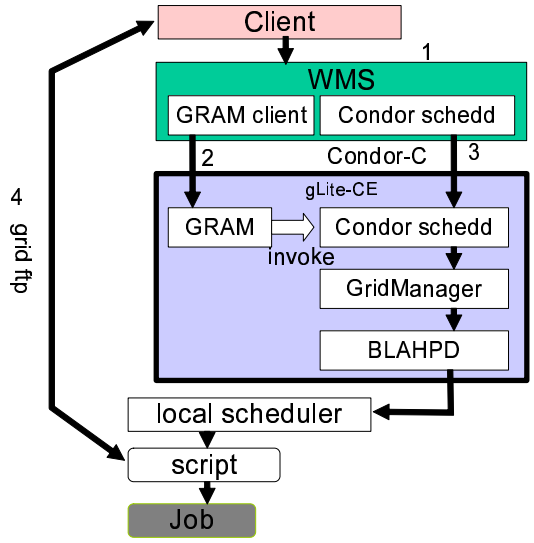

Figure 6. Job Execution in gLite CE.

4.1. Job execution with gLite-CE

GLite-CE uses a job submission protocol called Condor-C. Condor-C is a way to delegate job execution from one Condor schedd (scheduler daemon) to another Condor schedd. In gLite-CE, there is a Condor schedd on the node which is responsible for receiving job execution request from another schedd installed in the submission client. The schedd in gLite-CE execute the job using the local scheduler, such as PBS or LSF, via daemons called GridManager and BLAHPD, which will be described below. To invoke the schedd on for each user on gLite-CE node, the pre-WS GRAM protocol is used, to cover the aspect of grid security.

BLAHPD is a module to provide abstract interface for local batch queuing systems, such as PBS or LSF. The interface is text-based and simple to implement. It hides the details of each interface of the backend batch queuing system.

There are two notable characteristics for gLite-CE. One is that, it uses 'virtual user' as the local executing user account for daemons and jobs for each grid user. In more detail, it has a pool of virtual user accounts and allocates one of the pooled accounts when it gets a request from a grid user. This means that administrator does not have to create a local user account for each grid user at every site, as is done for Globus with gridmap files

Another point is that, the submission script passed to the local batch queuing system is not to directly exeucte the binary of the job submitted by the grid user, but rather a bootstrap; i.e., the script will access the client node and stage the real executable file and input data, execute it, and then send back the result to the client. The script expects to find a common supporting environment on the execution node, including GridFTP and several gLite specific commands and libraries.

Figure 6 shows the execution steps with gLite-CE.

1. Users submit their jobs from client node using submit command into the WMS.
2. WMS allocates a site to execute the job based on resource information obtained from BDII, invoke schedd on the site using the pre-WS GRAM protocol.
3. WMS submits the script described above into the schedd on CE using Condor-C protocol. The script will be passed to the backend queuing system via GridManager and BLAHPD.
4. The script downloads the executable and input files from the client using GridFTP, executes the executable, and uploads the result.

5. Interoperation Strategy

5.1. Security Infrastructure Interoperation

The security infrastructure is the base of all communication and other activities of the middleware stacks and its interoperability is crucial. Both of NAREGI Middleware Beta and gLite authenticate users and resources based on X.509 certificates issued by their certificate authorities. Operationally, both of their certificate authorities actually belong to PMA (Policy Management Authority) of IGTF (International Grid Trust Federation), enabling automatic interoperation of their certificates.

Another possible obstacle would be VO (Virtual Organization) management interoperability. Fortunately, NAREGI Middleware Beta is using the VOMS (Virtual Organization

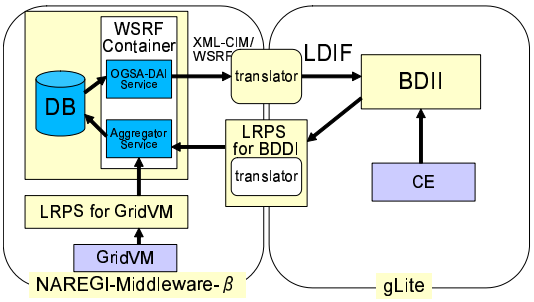

Figure 7. Information Service Interoperation.

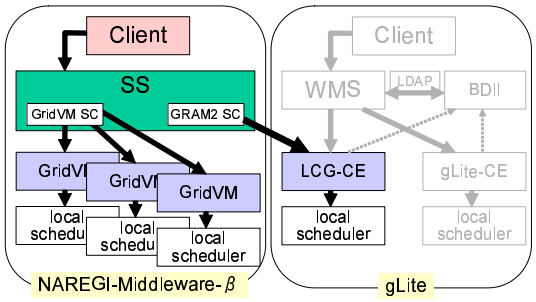

Figure 8. Job Submission from NAREGI Middleware Beta to gLite.

Membership Server) developed by gLite, effectively avoiding this possible obstacle. NAREGI middleware acutally involves additional layer of signing distinguishing the login session of the user of the grid from the actual workflow submission instances of each user, the latter requiring an extra layer of signing. Fortuantely, the certificate delegation employed by Globus GSI is not affected by this extra layer as it is merely another delegation step.

5.2. Information Service Interoperation

Since the information services for NAREGI Middleware Beta and gLite have different resource information schema (CIM vs. GLUE), different data format (CIM-XML vs. LDIF), and different data transfer protocols (OGSA-DAI vs. LDAP query), some intermediate modules are required to enable interoperation between these two information services, as shown in [18]. We developed two modules for this purpose, as shown in figure 7. The two modules shown in the figure is the intermediate modules. The upper one pulls information out from NAREGI Middleware Beta and pushes it into gLite, and the lower does the inverse, while translating the schema in between.

The upper componet is implemented as a stand alone translator module, which is activated periodically, retrieves CIM formatted information from Information Service, converts them into GLUE schema, formats them with LDIF, and stores them into BDII with LDAP protocol.

The lower component is implemented as a LPRS, which is one of the sub modules composing the NAREGI Information Service. It is also periodically activated, retrieves GLUE schema information from CIM with LDAP protocol from BDII, convert it into CIM schema. It compares them with the information previously retrieved and finds data that are updated since previous retrieval. If there are any updated data, the module 'notifies' Aggregator Service, which is another module composing the Information Service, with the WS-notification protocol. The Aggregator Service, in response to the notification, retrieves the updated information, sending retrieval message to the LPRS, and stores them to the backend database.

5.3. Job Submission Interoperation

There are two possible ways as shown in section 2 to realize job submission interoperation. We investigated the implementation of the two middleware stacks, and determined to employ a hybrid approach; we employ job manager level interoperation for job submission from NAREGI Middleware Beta to gLite, and broker level interoperation for job submission from gLite to NAREGI Middleware Beta.

6. Job Submission from NAREGI Middleware Beta to gLite

As mentioned above, we employed job manager level interoperation for this, because gLite LCG-CE is effectively equivalent with the well-known pre-WS GRAM from Globus Toolkit, so the support of the protocol was straightfoward. Figure 8 shows the overview of the implementation. Here we setup a path to the LCG-CE in parallel with the path to the GridVM.

In SS, there are modules called SC (Service Container) that abstract the external modules such as GridVM. The SCs act as abstract proxies for external modules. We implemented a SC for LCG-CE for interoperation. An SS has to know which SC is to be used for each specific site, based on the running modules there. IS provides information for this.

NAREGI Middleware Beta assumes that all the lower modules can handle advance reservation request and performs scheduling based on reservations, [2] while LCG-CE

2 This restriction will be lifted for the official version 1.0 release, allowing mixture of reserved/non-reserved grid jobs, as well as non-grid jobs submitted from the side

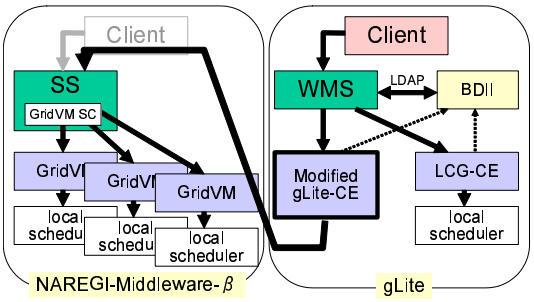

Figure 9. Job Submission from gLite to NAREGI Middleware Beta.

does not support advance reservation. We implemented a 'facade' advance reservation interface on the SC for LCG-CE. The SC always answers 'success' for requests for advance reservation on the resource, while it actually just submits jobs to the ordinary queue. With this implementation, we could accomplish the interoperability without changing the core logics in the SS.

7. Job Submission from gLite to NAREGI Middleware Beta

7.1. The Design

The gLite job submission path is designed to be easy to modify by the the use of the BLAHPD, the mechanism e employed in order to accomplish the interoperability. We implemented a BLAHPD that acts as a bridge to the NAREGI Middleware Beta. The BLAHPD receives a job submission request from schedd and instead of forwarding it to backend batch queuing system, which is the behavior of the normal BLAHPD, it forwords the job into NAREGI Middleware Beta, as shown in figure 9

The modified BLAHPD uses certificates delegated through the gLite invocation path to submit jobs to NAREGI Middleware Beta. The Condor schedd module in gLite-CE creates certificate files and passes the paths of the files to the modified BLAHPD, embedding them as environment variables. The BLAHPD reads the specified files and puts them into the MyProxy server. The client library for NAREGI Middleware Beta uses the certificate for further operation, i.e., submission and monitoring of the job.

7.2. Implementation of NAREGI BLAHPD

BLAHP is one of the variations of *GAHP* [7] which is a text protocol used in Condor to communicate with other grid middleware components, such as Globus Toolkit GRAM. The GAHP defines the format of requests and replies, how they interact, and how data should be marshaled. BLAHP uses GAHP as the transport protocol and adds concrete command set to be used within the protocol. We show the command set in figure 1.

Name	BLAH_JOB_SUBMIT
Request	`BLAH_JOB_CREATE <req id> <job classad>`
Return	`<S|E>`
Result	`<req id> <S|F> <error string>` `<job handle>`
Desc.	Creates a job handle for specified job, and starts it. Returns the job handle.

Name	BLAH_JOB_STATUS
Request	`BLAH_JOB_STATUS <req id> <job handle>`
Return	`<S|E>`
Result	`<req id> <S|F> <error string>` `<job status int> <result classad>`
Desc.	Returns status of a job which is specified by the job handle as a ClassAd

Name	BLAH_JOB_CANCEL
Request	`BLAH_JOB_CANCEL <req id> <job handle>`
Return	`<S|E>`
Result	`<req id> <S|F> <error string>`
Desc.	Cancels a job specified by the job handle, and releases resources related to the job.

Table 1. BLAHP Command Set

The BLAHP command set is similar to that of the UNICORE GAHP, which was implemented by ourselves in our previous work [19]. We implemented BLAHP for NAREGI Middleware Beta modifying the UNICORE GAHP server, which is written in Java. For job submission to NAREGI Middleware Beta, we used Java client API.

7.3. Issues Encountered

7.3.1. Virtual User Mapping As mentioned in 4.1, gLite CE uses virtual users as the executing UNIX user accounts, while NAREGI Middleware Beta uses the traditional grid user to local user Id mapping via Globus gridmap files, and does not support the concept of virtual users. This causes several problematic issues, such as staging files including the executables, onto compute nodes that are administered by the NAREGI Middleware (beta).

In NAREGI Middleware Beta, file staging are initiated by the SS, using *globus-url-copy* underneath to stage files from client sites to target sites, which is managed by GridVMs. GridFTP usually uses the Globus grid map file to map certificate DN into UNIX user and for proper authorization. The problem occurs when SS tries to stage files that belong to a virtual user using certificate. The grid map file on the

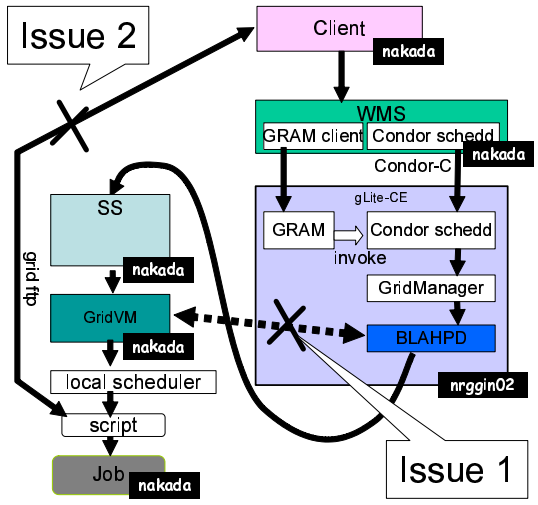

Figure 10. Details of gLite-NAREGI Middleware Beta job submission.

CE node, which is also acting as NAREGI Middleware Beta client node, will naturally not include thin mapping. Figure 10 shows the details of the job submission from gLite to NAREGI Middleware Beta. The small tags attached to the components at right-lower corner denote the UNIX user name recognized therein. Here, note that, in the glite-CE, the original user "nakada" no longer is valid, and replaced for a virtual user "nrggin02", which naturally is not recognized. This causes GridFTP failure between the gLite-CE and the GridVM.

There are several possible strategies to cope with this problem. The ideal way to handle this is to dynamically update the grid map file on the CE node with the user DN and the allocated virtual user. Less ideal but easer way is to dynamically create a directory that can be world accessible, and copy the contents of working directory to and from the directory. While world accessible directory may cause serious security problem in general, only a tiny script gLite could generates would use this directory in reality, on a time-limited basis—once the file is transferred, a copy would be made to an appropriate private directory, making the window of security bleach be very small. Because of limited time, we employed the latter strategy, but in the future we will likely employ the former approach.

7.3.2. Limitation in the number of possible delegations in GridFTP During the experiment, we found a 'bug' in the GridFTP server implementation in the Globus Toolkit, which is used in both middleware stacks. The bug effectively limits the number of times a certificates can be delegated to just eight.

Client \ Server	gLite-CE	LCG-CE	NAREGI
gLite	250	284	487
NAREGI	N/A	134	66

Table 2. Elapsed Time(sec.)

Since both middleware stacks leverage certificate delegation , and in particular job submission path from gLite to NAREGI Middleware Beta goes through both middleware stacks, and as a result, NAREGI Middleware Beta executable nodes receive certificates delegated more than 8 times, resulting in a failure of GridFTP between the gLite client node and NAREGI Middleware Beta executable nodes. The 'Issue 2' tag in figure 10 denotes the place where this happens. To cope with this problem, we had to fix GridFTP so that it accepts such certificates with long delegation paths.

8. Experimental Evaluation

To test the viability and the performance of the interoperation modules, we performed a controlled experiement within a testbed created locally within NAREGI for interoperation testing.

We set up groups of nodes with gLite version 3.0.2 and NAREGI Middleware Beta with the interoperation module we developed properly installed and submitted jobs from one to another. We installed the modules of two middleware stacks so that each module occupies one host. All the hosts have the same basic configuration; i.e. dual Pentium 4 Xeon 3GHz, 1G byte Memory, RedHat 8 Linux, interconnected with a 100 base/TX network. GLite uses TORQUE and NAREGI Middleware Beta uses PBS professional as backend queuing systems.

We have measured the job submission latency for a job that executes 'hostname' command on the executing server and returns a result string. Since the job execution itself is negligible, the latency directly shows the middleware overhead. We also measured job submission confined within the respective middleware stacks for comparison.

Table 2 shows the result of the experiment. The experiment had been repeated 10 times and the average time is shown here. Please note that both of the middlewares require polling to detect the completion of job execution. Since we set up the polling interval as 10 seconds, the numbers shown in the table have error within 10 seconds.

From the table, we can see that the case from gLite to NAREGI Middleware Beta is the slowest likely due to the long execution path that goes through both brokering systems. We also observe that all submission times

are comparable gLite - LCG-CE submission, which is deployed widely in production and shown to handle more than 100,000 jobs per day. In fact, as one can observe, NAREGI - LCG-CE submission is faster, despite the interoperation. As such, we claim that our interoperation design does not add overhead that would render the system unusable in practice in terms of performance — a promising performance number for true global grid interoperation.

9. Conclusion

We performed interoperation experiments between NAREGI Middleware Beta and EGEE gLite and confirmed the followings: 1) there are no issues regarding the certificates and VO management layer 2) in information services, there are substantial differences as note but this can be alleviated with approrpriate schema/format conversion, 3) in job submission, by careful design that would exploit the properties of respective middleware and devising an appropriate architecture for each path in the job submission amongst the components, we confirmed that it is possible to submit jobs from one to another with realistic performance.

We have several future work including:

- The experiments shown in this paper was performed within the NAREGI site in a controlled fashon so as to exclude the effects come from real distributed environment, such as long latency or poor network stability. The next step is to design and perform experiments with Japan and European sites using large-scale (pre) production infrastructure resources.

- The gLite assumes that all the jobs executing host can access to the gLite executing environment, i.e., several libraries and commands, such as GridFTP, Also it assumes that an executing host would have direct access to the client node, which is not usual in a restricted grid environment. While for experimentation we installed gLite executing environment on all the NAREGI hosts for this experiment, we do not believe that such dependence on the availability of middleware stack specific commands would be feasible for multi-tier interoperational settings. We need to investigate some other way to avoid the installation of gLite executing environment on each execution node of NAREGI Middleware Beta, such as the use of virtual machines to ship the entire operational environment in a closed, self-consistent fashion.

Acknowledgement

A part of this research was supported by a grant from the Ministry of Education, Sports, Culture, Science, and Technology (MEXT) of Japan through the NAREGI (National Research Grid Initiative) Project.

References

[1] Common Information Model. http://www.dmtf.org/standards/cim/.
[2] Condor. http://www.cs.wisc.edu/condor/.
[3] Distributed Management Task Force. http://www.dmtf.org/search.
[4] EGEE: Enabling Grids for E-SciencE. http://www.eu-egee.org/.
[5] EGEE Middleware Architecture and Planning. Technical Report DJRA1.4, EU Deliverables.
[6] gLite: Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/.
[7] Globus ascii helper protocol. http://www.cs.wisc.edu/condor/gahp/.
[8] GLUE Schema. http://glueschema.forge.cnaf.infn.it/.
[9] Job Submission Description Language (JSDL) Specification. Open Grid Forum, GFD.56.
[10] Open Grid Forum. http://www.ogf.org.
[11] Open Grid Forum Grid Interoperation Now Community Group. https://forge.gridforum.org/sf/projects/gin.
[12] The OGSA-DAI Project. http://www.ogsadai.org.uk/.
[13] Unicore. http://www.unicore.org/.
[14] Web Services Resource Framework. http://www.oasis-open.org/committees/wsrf.
[15] K. Czajkowski, I. Foster, C. Kesselman, N. Karonis, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.
[16] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pages 2–13, 2005.
[17] S. Matsuoka, M. Hatanaka, Y. Nakano, Y. Iguchi, T. Ohno, K. Saga, and H. Nakada. Design and implementation of naregi superscheduler based on the ogsa architecture. 21(4):521–528, July 2006.
[18] M.Gønager , et.al . LCG and ARC middleware interoperability. In *Proceedings of CHEP 2006*, number 348, 2006.
[19] H. Nakada, J. Frey, M. Yamada, Y. Itou, Y. Nakano, and S. Matsuoka. "design and implementation of condor-unicore bridge". In *"Proceedings of HPC ASIA 2005"*, pages 307–314, 2005.
[20] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proc. of HPDC-7*, 1998.
[21] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). RFC 2251.