PROCEEDINGS OF SCIENCE

SAGA-based application to use resources on different Grids

Yutaka Kawai*, **Go Iwai, Takashi Sasaki and Yoshiyuki Watase**
High Energy Accelerator Research Organization (KEK)
E-mail: yutaka.kawai@kek.jp, go.iwai@kek.jp,
takashi.sasaki@kek.jp, yoshiyuki.watase@kek.jp

This paper describes practical applications using resources on different kinds of Grid middleware. At the KEK Computing Research Center, many jobs must be submitted for physics simulations involving large number of data files from physics experiments. The available resources of the various Grids should be used in a cooperative way, but specialized knowledge is currently required to use each Grid. Our solution is using SAGA (Simple API for Grid Applications), which provides a unified interface that conceals the differences among the different kinds of Grid middleware. We developed SAGA adaptors for job execution, file management, and catalog services. The job adaptors we created are applied to each kind of Grid middleware: NAREGI, PBSPro, and Torque. The file adaptors support the Data Grids: iRODS and Gfarm. The replica adaptor currently in use is for the catalog service: RNS (Resource Namespace Service) and iRODS. SAGA with the adaptors allows us to utilize the various Grid resources along with local resources without any concerns about the underlying middleware. Technical details and sample applications are described in this paper.

The International Symposium on Grids and Clouds and the Open Grid Forum - ISGC2011,
March 25-30, 2011
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC 2011 & OGF 31)021

1. Introduction

Currently we have a number of computing systems and storage resources in different kinds of Grid environments at KEK. Each Grid relies on advanced middleware to interface between resources and applications[8]. We use different Grid resources with various kinds of middleware to ensure compatibility with user applications that are designed for their own Grid environments. Our primary objective is to demonstrate unified methods for the development and execution of applications by users who are the application developers in various domestic communities. Scientists who are geographically distributed require standardized access to distributed storage resources without specialized configurations for each kind of middleware. We are studying APIs for multiple kinds of Grid middleware as a part of the RENKEI (REsources liNKage for E-scIence) project[17] to help end users efficiently use both Grid and non-Grid resources. RENKEI is a research and development project for middleware technology to utilize computing resources, files, DBs, and similar resources in heterogeneous environments. The RENKEI project also has a subproject to develop a file catalogue system using RNS (Resource Namespace Service)[11].

This paper describes how to use resources distributed in different Grids. The example of this paper uses applications based on SAGA (A Simple API for Grid Applications)[7, 2] to span the different Grid environments. Jobs are submitted to three kinds of Grids: NAREGI (National Research Grid Initiative)[12, 13], PBSPro (Portable Batch System Professional Edition)[15], and Torque[21]. To manage files, we use two kinds of Data Grids: iRODS (The integrated Rule-Oriented Data System)[4, 16] and Gfarm (Grid data farm)[1]. The information about the physical file locations in our environment is managed as metadata entries in RNS.

This paper gives an overview of SAGA in Section 2. Section 3 describes how to submit jobs with SAGA job adaptors. Section 4 describes how we use SAGA file adaptors and replica adaptors to manage files under different Grid environments. Then, our results and future work are covered leading to our conclusions.

2. SAGA Overview

Using Data Grids involves using their specialized commands and rules to access their files. The differences between these commands and rules can cause problems for application developers. SAGA provides a unified interface that conceals the differences among the different middleware infrastructures. The abstraction layer of SAGA is positioned as a bridge among the various kinds of Data Grids. Once a SAGA adaptor for each kind of middleware has been prepared, the application developers only need the functional API without worrying about the specific features of the middleware. We are developing the SAGA adaptors shown Table 1, in compliance with the SAGA specification, Version 1.0[2], as standardized[7] within the OGF[14].

We can easily use any type of Grid resources as one resource if the appropriate SAGA adaptors are available. The SAGA community has released several SAGA core implementations[20]. The current SAGA C++ implementation supports wide range of Grids[19].

2.1 SAGA APIs

Python, C++ and Java APIs are currently available as the functional SAGA API. The APIs are

Table 1: SAGA Adaptors under development in KEK.

Name	Full Name	Middleware
SNA	SAGA NAREGI Adaptor	NAREGI
STA	SAGA Torque Adaptor	Torque
SPA	SAGA PBSPro Adaptor	PBSPro
SIA	SAGA iRODS Adaptor	iRODS
SGFA	SAGA Gfarm Adaptor	Gfarm
SRA	SAGA RNS Adaptor	RNS

used for job modules, file modules, and replica modules.

2.1.1 SAGA APIs for Job Modules

Application developers can use when invoking the APIs to submit jobs to the specified middleware. Application users need only specify the scheme to switch to different Grid middleware with the same job description.

2.1.2 SAGA APIs for File Modules

Application developers can use the APIs to use the file system via SAGA. Application users need only specify the scheme and logical path to switch to some other file system middleware.

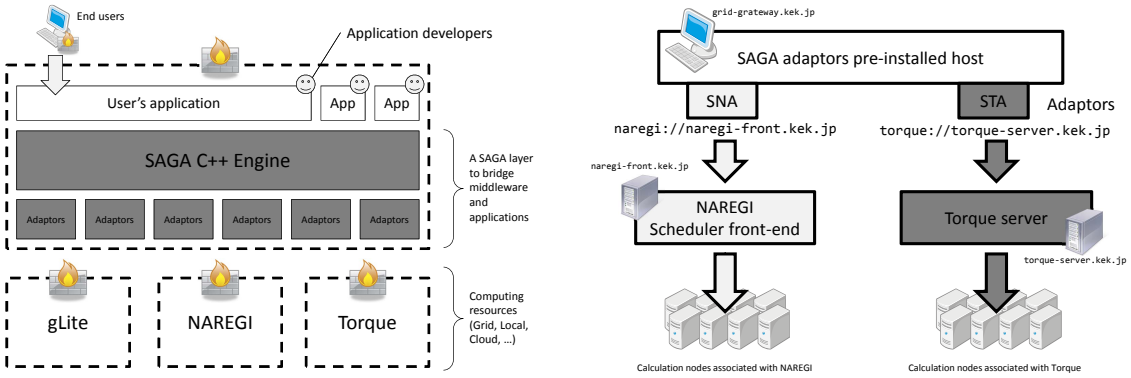
2.1.3 SAGA APIs for Replica Modules

We also need to control the RNS application via SAGA. The logical file and the logical directory in the SAGA replica package allow us to handle the required metadata. Application developers can use the APIs to handle metadata for logical files and directories.

3. Job Submission

We already presented some SAGA-based applications using job submissions for various Grid environments[6]. Figure 1(a) shows a more detailed architecture with SAGA. The SAGA layer is located between the "End users" and the various kinds of computing resources. Even if firewalls exist between the computing resources and the higher-level components (i.e. SAGA, applications, or end users), the applications can still use the resources from all of the Grids through the SAGA adaptors.

We developed SAGA adaptors for NAREGI (SNA: SAGA NAREGI Adaptor), for PBSPro (SPA: SAGA PBSPro Adaptor), and for Torque (STA: SAGA Torque Adaptor). The commands of PBSPro and Torque seem to be similar but we need each different adaptor for PBSPro and Torque. That is because several definitions of the command options are different. Those adaptors comply with the SAGA specification version 1.0[2] standardized through the OGF. Our demonstration works in a SAGA environment with SNA and STA. The SAGA library, SNA, and STA are all installed on a host server that is called the "SAGA adaptor host". All of the client applications



(a) SAGA-based user environment.

(b) Workflow diagram in the user environment based on SAGA.

Figure 1: Multi-middleware platform based on SAGA

```
import saga
import sys

argvs = sys.argv
argc = len(argvs)

# Create a Job Description
js_url = saga.url(argvs[1])
job_caht = js_url.get_host()
job_service = saga.job.service(js_url)
job_desc = saga.job.description()
job_desc.executable = './job_example.sh'
job_desc.working_directory = '$HOME/work_dir'
job_desc.candidate_hosts = job_caht
# Submit a job
my_job = job_service.create_job(job_desc)
my_job.run()
```

Figure 2: Job execution example (sample.py) using python interface.

should also be installed on the SAGA adaptor host. Figure 1(b) shows the workflow diagram for our demonstration.

The same application can be executed by either NAREGI or Torque middleware. Figure 2 shows sample code that submits a job using the SAGA Python API. In this case, the job description is simply defined in the code. The application developer can separate the job description from the code if necessary. In this example, the users only need to specify the pair of a job service and a hostname as the argument. For example, here is a command to submit a job to NAREGI:

```
$ python sample.py naregi://naregi-front.kek.jp
```

The correspinding command to submit the job to Torque is:

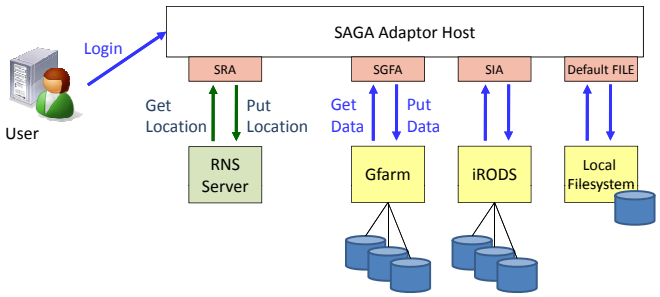


Figure 3: Workflow diagram in the user environment for file management.

```
$ python sample.py torque://torque-server.kek.jp
```

There is no need to change the application itself, as shown in this example. Application developers do not need to worry about any compatibility issues among the middleware.

4. File Management and Catalog Service

Our implementation requires masking the schemes of the different kinds of Data Grids with SAGA. SAGA-based applications can work for the various Data Grid environments[9]. The prototype adaptors we developed work in a SAGA environment with iRODS, Gfarm, a local file system, and RNS. Therefore, we used SIA, SGFA, and SRA. The client applications for iRODS and Gfarm should also be installed on the SAGA Adapter Host. Figure 3 shows the workflow for our experiments.

The client application should get the information about the physical file locations from the RNS server via SRA before accessing the Data Grids. When the application stores files on the Data Grids, the application should send the information about the file location to the RNS server after storing the files.

4.1 Example

There are many analyzed test results to be shared between KEK and other research organizations. For large data files, it is helpful to share the image files by dividing them into pieces, storing them on different kinds of Data Grids, and registering the physical file information into RNS. SAGA-based applications can handle distributed files with RNS in heterogeneous Data Grids[10].

We created a sample SAGA application, which retrieves the physical locations of the distributed files and concatenates the identified files. We registered an RNS virtual directory and three junctions:

```
rns://sg01.cc.kek.jp/100780001
                      ├──100780001_aa
                      ├──100780001_ab
                      └──100780001_ac
```

```
<?xml version="1.0" encoding="UTF−8"?>
<file xmlns="http://kek.jp/rns/test">
  <rnskv key="file_num" xmlns="">3</rnskv>
  <rnskv key="file_data_0" xmlns="">100780001_aa</rnskv>
  <rnskv key="file_data_1" xmlns="">100780001_ab</rnskv>
  <rnskv key="file_data_2" xmlns="">100780001_ac</rnskv>
</file>
```

Figure 4: The example of the attribute definition

Table 2: Physical resource locations of the divided example files.

File System	URL
iRODS	irods://sg03.cc.kek.jp/tempZone/home/kawai/bubble/100780001_aa
Gfarm	gfarm://sg07.cc.kek.jp/kawai/bubble/100780001_ab
Local File System	file://sg01.cc.kek.jp/home/kawai/bubble/100780001_ac

The RNS path-name of the virtual directory is required as an argument for the application. The hostname "sg01.cc.kek.jp" is the RNS server in our environment. Therefore, the RNS path-name of the argument becomes "rns://sg01.cc.kek.jp/100780001". We can specify the metadata of the RNS virtual directory, 100780001, as shown in Figure 4. Table 2 shows the physical locations of the divided files. Each URL for a physical location is specified in the format of a SAGA URL compliant with RFC1630[18].

If we use an image file that is divided and stored in different Data Grids, we can use the "display" command of ImageMagick[3] to display the output of the SAGA application. Here is the "img_cat_rns" command:

```
$ img_cat_rns rns://sg01.cc.kek.jp/100780001 | display
```

The SAGA application retrieves the locations of the distributed files from the RNS server, combines the pieces and displays them as an image. There is no need to change the application if we use different kinds of Data Grids. All we need to do is change the RNS entries and the metadata. Application developers do not need to worry about any incompatibilities among the different kinds of middleware.

5. Results

The SAGA based python program handles the job submissions to different Grids. The whole process of this workflow was described in a simple python program that is easily readable by the end users. For application developers, this provides a convenient environment for debugging and tuning applications.

Our sample SAGA application retrieves the location information for distributed files on the different kinds of Data Grids by using RNS to access the files. This simple approach is easily understood by the end users because the coding method is similar to a traditional program accessing

a local file system and it only requires some simple key-value operations. The file location information is managed in the RNS system without modifying the source code of the application. For application developers, this provides a convenient environment for debugging and tuning applications, because they do not need to worry about the differences among the middleware.

6. Future Work

We plan to implement SAGA-based applications/tools in the future. UGAPI (The Universal Grid API[5]) is one of examples. We can enhance UGAPI with the above SAGA adaptors we developed. UGAPI offers typical functionalities required by end users as submitting and managing jobs, or file access both for the different distributed computing infrastructures including local batch queuing systems. We will create UGAPI-based prototypes for the physics simulation (i.e. particle therapy simulation) executing the large scale calculation using the different infrastructures at the same time.

7. Conclusion

We have shown how our SAGA based applications work using resources on the different kinds of Grid middleware. We developed SAGA job adaptors, file adaptors, and replica adaptors. We showed that there is no need to change an application to submit jobs and to access files on the different Grid environments with SAGA by using the adaptors. Therefore, application users can use the different Grid resources as well as local resources without any concerns about the underlying middleware. This approach can be used in a Grid computing system to handle distributed Grid resources.

8. Acknowledgment

It is a pleasure to acknowledge the SAGA developer team led by Shantenu Jha and Andre Merzky for their valuable suggestions and support. We also would like to thank the RNS implementation development team led by Hideo Matsuda of Osaka University and Osamu Tatebe of the University of Tsukuba. The iRODS technical support from Adil Hasan of the University of Liverpool is gratefully acknowledged.

References

[1] Gfarm – Grid Data Farm. Online. `http://datafarm.apgrid.org/index.en.html`.

[2] Goodale, Tom, et al. SAGA - v1.0 Specification (GFD-R-P.90). Technical report, SAGA-CORE-WG, 2008. `http://www.ggf.org/documents/GFD.90.pdf`.

[3] ImageMagick. Online. `http://www.imagemagick.org/script/index.php`.

[4] iRODS – the Integrated Rule-Oriented Data System. Online. `http://www.irods.org`.

[5] G. Iwai, Y. Kawai, T. Sasaki, and Y. Watase. A Development of Lightweight Grid Interface. In *Poster. the International Conference on Computing in High Energy and Nuclear Physics, CHEP 2010*, Taipei, Taiwan, Oct. 2010.

[6] G. Iwai, Y. Kawai, T. Sasaki, and Y. Watase. SAGA-based user environment for distributed computing resources: A universal Grid solution over multi-middleware infrastructures. In *Proc. International Conference on Computational Science, ICCS 2010.*, pages 1539–1545, Amsterdam, Netherlands, May 2010.

[7] S. Jha, H. Kaiser, Y. El Khamra, and O. Weidner. Design and Implementation of Network Performance Aware Applications Using SAGA and Cactus. In *Proc. The 3rd IEEE Conference on eScience2007 and Grid Computing.*, pages 143–150, Bangalore, India, Dec. 2007.

[8] B. Jones. EGEE - a worldwide Grid infrastructure, August 2005. The 19th International Congress of the European Federation for Medical Informatics (MIE), Geneva. `http://egee-intranet.web.cern.ch/egee-intranet/NA1/presentations/ppt-fbm/2005/MIE-2005.ppt`.

[9] Y. Kawai, G. Iwai, T. Sasaki, and Y. Watase. SAGA-based File Access Application over Multi-Filesystem Middleware. In *Proc. Challenges of Large Applications in Distributed Environments, CLADE 2010*, pages 622–626 (CD–ROM), Chicago, US, June 2010.

[10] Y. Kawai, G. Iwai, T. Sasaki, and Y. Watase. Managing distributed files with RNS in heterogeneous Data Grids. In *Proc. the 11th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2011*, California, US, May 2011.

[11] H. Matsuda. File Catalog Development in Japan e-Science Project. Technical report, GFS-WG, 2008. `http://www.ogf.org/OGF24/materials/1403/OGF24-GFS-matsuda.pdf`.

[12] S. Matsuoka, S. Shimojo, M. Aoyagi, S. Sekiguchi, H. Usami, and K. Miura. Japanese Computational Grid Research Project: NAREGI. *Proceedings of the IEEE*, 93(3):522–533, March 2005.

[13] NAREGI – NAtional REsearch Grid Initiative. Online. `http://www.naregi.org/index_e.html`.

[14] OGF – Open Grid Forum. Online. `http://www.ogf.org/`.

[15] PBS Professional. Online. `http://www.pbsworks.com/Product.aspx?id=1`.

[16] A. Rajasekar, M. Wan, R. Moore, and W. Schroeder. A Prototype Rule-based Distributed Data Management System. In *Proc. HPDC workshop on "Next Generation Distributed Data Management"*, Paris, France, May 2006.

[17] RENKEI – REsources liNKage for E-scIence. Online. `http://www.e-sciren.org/index-e.html`.

[18] RFC1630 – Universal Resource Identifiers in WWW. Online. `http://www.ietf.org/rfc/rfc1630.txt`.

[19] Available adaptors in SAGA. Online. `http://saga.cct.lsu.edu/software/cpp/adaptors`.

[20] SAGA: A Simple API for Grid Applications. Online. `http://saga.cct.lsu.edu/`.

[21] TORQUE Resource Manager. Online. `http://www.clusterresources.com/products/torque-resource-manager.php`.