PROCEEDINGS OF SCIENCE

Universal Grid User Interface (UGI) for Multiple Grids and Clouds

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

This paper describes a practical application of a Universal Grid User Interface (UGI) to control resources with different kinds of middleware. Today, international scientific collaboration requires shared hardware and software resources provided by various kinds of Grid and Cloud middleware. We have designed and implemented a UGI that can provide a seamless environment for end users to use such remote resources (Grid or Cloud resources) with their local resources. The UGI functions include of job handling, manipulating files, general file cataloging, and monitoring jobs. UGI includes the SAGA (Simple API for Grid Applications) architecture and external components that are not supported by SAGA. Our prototype UGI implementation provides a Python API, a command line interface, and a Web interface. We show an example of a UGI application with a Web-based user interface for Particle Therapy Simulation (PTSim).

The International Symposium on Grids and Clouds (ISGC) 2012,
Febraury 26 - March 2, 2012
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC 2012)033

1. Introduction

Recent scientific challenges require worldwide collaboration of researchers sharing their resources, such as computing power, large distributed data, software, and knowledge. In the last 10 years, different nations or regions have developed and deployed different middleware infrastructures for e-science. Actually, 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[1]. We use different Grid resources with various kinds of middleware to ensure compatibility with user applications that are designed for their own Grid environments. We are studying APIs for multiple kinds of Grid middleware as a part of the RENKEI (REsources liNKage for E-scIence) project[2] to help end users efficiently use both Grid and non-Grid resources. 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.

In order to share scientific resources among collaboration members, we have to cope with different user interface to these Grid middleware. We adopted SAGA (A Simple API for Grid Applications)[3, 4] to span the different Grid environments. SAGA aims to address this heterogeneity and currently provides its working implementations in C++ and Java. Jobs are submitted to several kinds of Grids and local batch systems : NAREGI (National Research Grid Initiative)[5, 6], gLite[7], Globus[8], PBSPro (Portable Batch System Professional Edition)[9], and Torque[10]. To manage files, we use two kinds of Data Grids: iRODS (The integrated Rule-Oriented Data System)[11, 12] and Gfarm (Grid data farm)[13]. The information about the physical file locations in our environment is managed as metadata entries in RNS (Resource Namespace Service)[14, 15]. The SAGA API facilitates application developers to build real applications used by the end user of scientists. Our developments of such adaptors enabled to show how to use resources distributed in different Grids [16] and solves several problems [17, 18, 19].

We designed and developed a higher level user interface: Universal Grid user Interface(UGI) which consists of a Python API and command line interface. UGI is implemented based on SAGA and also provides supplemental and extended functionalities that are not given by SAGA. UGI-based Web interface with the various functionalities can share Grid/Cloud resources as well as local ones.

This paper gives an overview of SAGA in Section 2. Section 3 describes the UGI design concept. The implementation is shown in Section 4. In section 5, a practical application is described for the simulation of particle therapy using Geant4 [20, 21, 22].

2. SAGA Overview

Using Grid systems 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 Grids. Once a SAGA adaptor for each kind of middleware has been prepared, the application

Table 1: SAGA Adaptors developed 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

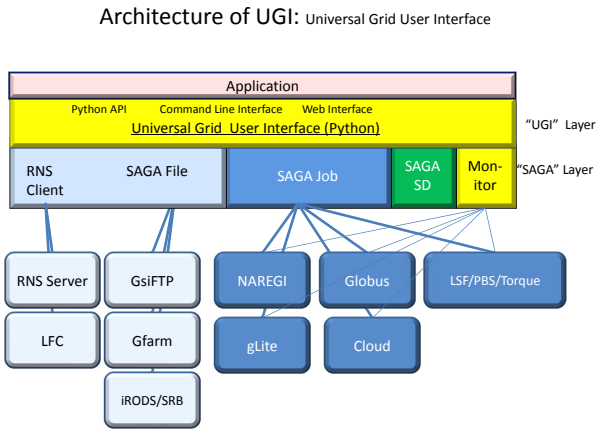

Figure 1: Architecture of Universal Grid Interface

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[4], as standardized[3] within the OGF[23].

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[24]. The current SAGA C++ implementation supports a wide range of Grids[25].

3. UGI Design

The motivation of UGI design is to hide the complex treatment of grid middleware from end user scientists and to provide them with seamless access to local and Grid/Cloud. Figure 1 shows the architecture of the software building blocks. The interfaces consist of Python API, Command Line, and Web Interface. The latter two are written using the former API. The reason for adopting Python for the API is its popularity with the scientists for quick development of their field applications. The Python API layer is the upper layer of the SAGA and written with the Python language binding of SAGA C++ and an additional Python wrapper for the command line interfaces of various clients as described in the next section. It is also an important aspect to be able to maintain and expand easily the API for future middleware development.

The current design of UGI functionality consists of job handling, file manipulation, general file catalog, and monitoring. It runs on a host which is installed with SAGA C++ core/adaptors and client software necessary to use the Grid middleware and GridFTP server.

For the job handling, the UGI provides multiple job submission to various Grid middleware infrastructures and local batch system at the same time using SAGA job submission scripts. The job load sharing can be set according to the resource availability. The real end user application handles the results and data files according to their work flow design, such as chaining jobs, post processing, and graphic display.

UGI provides file manipulation functions such as copy, remove, transfer, and catalog registration. These functions accept various file storage protocol: file(local file), gsiftp, gfarm, and iRODS within the same API. The file catalog is a major facility for sharing large distributed scientific data. For sharing files among different Grid middleware, a middleware independent file catalog system is required. The UGI adopted OGF standard RNS as the file catalog. We collaborated with Osaka University and University of Tsukuba to implement the RNS as general file catalog system. It provides a tree structure of the name space with a virtual directory and junction. Each junction has a valid WSAddressing[26] EPR (Endpoint Reference). Both virtual directories and junctions can contain metadata on the file or directory. The metadata can be queried to find appropriate files to use from a large scale scientific data.

For the monitoring of a large number of jobs dispatched in different kinds of Grid environments, we introduced a light weight database. The database can store not only job status, but also job related information such as job parameters, analysis conditions, output file location, etc. Usually the status transition of jobs submitted in a Grid middleware is delayed due to the propagation time from the middleware to the client. In the UGI, the dispatched job can update the database by itself through RPC mechanism.

4. UGI Implementation

4.1 Job Handling

For a single job submission to Grid middleware, it is straight forward to write a script using the Python binding of SAGA C++ scheme. We introduced a task python object for multiple job (parametric job) submission to a Grid middleware resource in different sites. The attributes of the object include middleware name to be submitted, site name , number of jobs sharing load, path of job script written in SAGA Python binding. The Python API for task submission is: `ugi.job.submit(list_of_task)` which gives simultaneous job/task submission to different kinds of Grid middleware. A typical command line interface is: ugi-job-muitiplejob-submit middleware, site, njobs, path with arguments following the above task attributes.

4.2 File Manipulation

Each Grid middleware has its own file storage system with its specific protocol for access. For multiple Grid environments, we need to hide these differences as much as possible. Using SAGA mechanism, the UGI provides uniform user interfaces. The available SAGA file adaptors are for file protocols: local file, gsiftp, gfarm, and irods. A typical example of API is a file listing under a

directory URL: `ugi.file.ls(url,options)` which returns a list object of file names. The url accepts, `file://...`, `gsiftp://...`, `gfarm://...`, `irods://...` with the option: `-l` for long listing. It also accept RNS catalog tree path like `/rns/kek/ilc`. For files registered in RNS, there are additional options to get information of the registered files: `-u` (get url of file), `-t` (get transfer url), and `-query` (metadata query). File transfer or copy is an important function in the file manipulation. The SAGA scheme cannot support direct file copy between different storage protocols. The UGI API: `ugi.file.copy()` accepts different protocols for source and destination urls by wrapping the proper commands to copy a file to/from the local file storage. The RNS server running on a separate host stores the file catalog in a database(the current implementation uses derby: JavaDB) . The UGI gets access to the catalog tree through FUSE [27] mount mechanism to avoid the overhead of the Java client VM. The UGI API for catalog manipulation are such as `ugi.file.register()`, `ugi.file.unregister()`, `ugi.file.replicate()`, and `ugi.file.transfer-register()`. UGI has command line interfaces corresponding to all of the API functionalities.

4.3 Monitoring

A sqlite3 database is used as a light weight database for storing the status of jobs dispatched to multiple Grid middleware as well as its job parameter and additional user defined job information such as output file location. The UGI provides a useful database access API and commands. The database information is updated by each job using these API or commands in the job script. In order to get access to the database from running jobs, the XML-RPC server/client mechanism is used. The server runs on the local host and waits for client commands invoked by the job script.

4.4 Security

User authentication/authorization may differ on each middleware. The UGI takes a practical approach to issue and/or register a proxy certificate using simple shell commands: `ugi-cert-init.sh` which is an integrated command containing the middleware commands.

5. Application

5.1 Particle Therapy Simulation

Geant4 [20, 21, 22] is a toolkit for the simulation of the passage of particles. Geant4 is also applicable to other general simulation of radiation processes. An application of the Geant4 simulation to the particle therapy using proton and ion beam is one of outreach activities from the high energy physics. We have developed a Particle Therapy Simulation(PTSim) [28, 29] system including graphical interface in collaboration with several medical particle therapy centers. A detailed simulation of the human body requires significant CPU time. It has been shown in previous studies that parallel processing is useful in getting results of a large number of events. Collected use cases in particle therapy simulation related on Grids are listed below:

Distributed job execution

It is effective to divide a simulation job into a lot of smaller jobs in order to gain a large number of event productions.

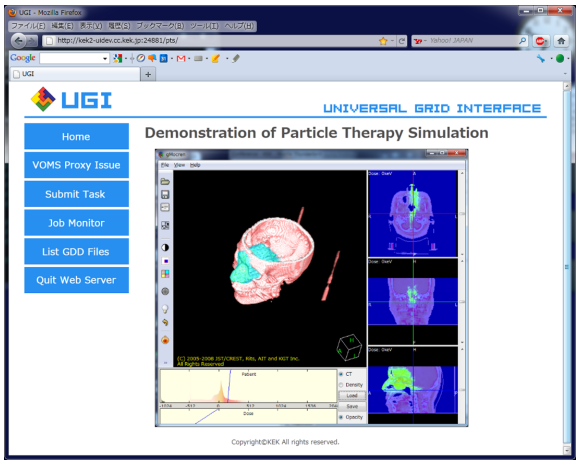

Figure 2: Application for Particle Therapy Simulation Work Bench

Secure job execution

Secure has two kinds of context in this case. Firstly, users who are working in hospital are typically restricted in accessing the Internet. Secondly, patient's personal information must be kept inaccessible to unauthorized personnel.

Spatial requirement and storage capability

Large scale data such as DICOM (Digital Imaging and Communication in Medicine) [30] which is a standard format for the patient data scanned by medical imaging system, taken from CT needs to be stored inside of hospital for the treatment planning.

5.2 PTSim Web infterface

An example of UGI application is a Web based user interface for PTSim. The simulation required several tens of millions of particle ray tracings in a human body model. It requires thousands of multiple jobs to be processed at a time to get the result in a reasonable elapsed time. The PTSim Web interface is developed in the Django framework which is convenient for Web development with the Python language. The built-in Django Web server runs on the local host. An end user desk top PC browses it as shown in Figure 2.

The available kinds of Grid middleware in the current testbed are NAREGI, Globus GRAM2, gLite for computing resources and GridFTP server, Gfarm storage and iRODS for storage system. After login and proxy registration for NAREGI, multiple jobs described in a specific task script are submitted to these kinds of middleware and the job status is updated in the database and displayed. Each job transfers their output file to a GridFTP server. These output files are processed on the user desktop PC for graphic display. These actions are done on the Web.

6. Conclusion

We developed UGI interface and a UGI-based Web application for particle therapy simulation. UGI is implemented based on SAGA and provides supplemental and extended functionalities that

are not given by SAGA. UGI provides API for applications and command line interfaces based on the API. The prototype of Web interface enables users to request most of their job operations. UGI also make it possible that non-Grid applications working on local resources are portably exported to distributed resources over the Grid resources.

This approach can improve usability of interfaces for e-Science and the particle therapy simulation.

7. Acknowledgment

The authors would like to thank members of Geant4 collaboration and members of PTSim development, especially, Prof.Takashi Akagi and Tomohiro Yamashita for using the DICOM data. We would also like to thank the full support from Project RENEKI for building next generation Cyber Science Infrastructure, funded by the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

References

[1] Bob 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.

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

[3] Shantenu Jha, Hartmut Kaiser, Yaakoub El Khamra, and Ole 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, December 2007.

[4] 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.

[5] Satoshi Matsuoka, Sinji Shimojo, Mutsumi Aoyagi, Satoshi Sekiguchi, Hitohide Usami, and Kenichi Miura. Japanese Computational Grid Research Project: NAREGI. *Proceedings of the IEEE*, 93(3):522–533, March 2005.

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

[7] gLite: Lightweight Middleware for Grid Computing. Online. http://glite.web.cern.ch/glite/.

[8] Globus Toolkit. Online. http://www.globus.org/toolkit/.

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

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

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

[12] Arcot Rajasekar, Mike Wan, Reagan Moore, and Wayne Schroeder. A Prototype Rule-based Distributed Data Management System. In *Proc. HPDC workshop on "Next Generation Distributed Data Management"*, Paris, France, May 2006.

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

[14] Manuel Pereira, Osamu Tatebe, et al. Resource namespace service specification (GFD-R-P.101). Technical report, GFS-WG, 2007. `http://www.ggf.org/documents/GFD.101.pdf`.

[15] Hideo 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`.

[16] Yutaka Kawai, Go Iwai, Takashi Sasaki, and Yoshiyuki Watase. SAGA-based application to use resources on different Grids. In *Proc. International Symposium on Grids and Clouds (ISGC), in series Proceedings of Science*, Taipei, Taiwan, March 2011.

[17] Go Iwai, Yutaka Kawai, Takashi Sasaki, and Yoshiyuki 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), in series Procedia Computer Science*, pages 1539–1545, Amsterdam, Netherlands, May 2010. ISSN: 1877-0509.

[18] Yutaka Kawai, Go Iwai, Takashi Sasaki, and Yoshiyuki 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)*, pages 494–503, California, US, May 2011. ISBN: 978-0769543956.

[19] Yutaka Kawai, Adil Hasan, Go Iwai, Takashi Sasaki, and Yoshiyuki Watase. A method for reliably managing files with RNS in multi Data Grids. In *Proc. International Conference on Computational Science (ICCS), in series Procedia Computer Science*, pages 412–421, Singapore, June 2011. ISSN: 1877-0509.

[20] Geant4. Online. `http://geant4.cern.ch/`.

[21] John Allison et al. Geant4 developments and applications. *IEEE Trans. Nucl. Sci.*, 53:270–278, February 2006.

[22] S. Agostinelli et al. GEANT4 – A simulation toolkit. *Nucl. Instrum. Meth.*, A506:250–303, July 2003.

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

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

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

[26] Web Services Addressing 1.0 – Core. Online. `http://www.w3.org/TR/ws-addr-core/`.

[27] FUSE: Filesystem in Userspace. Online. `http://fuse.sourceforge.net/`.

[28] Tsukasa Aso, Akinori Kimura, Satoru Kameoka, Kouichi Murakami, Takashi Sasaki, and Tomohiro Yamashita. GEANT4 Based Simulation Framework for Particle Therapy System. In *Proc. IEEE Nuclear Science Symposium Conference Record*, pages 2564–2567, Hawaii, US, November 2007.

[29] Takashi Sasaki and Satoshi Tanaka. Comprehensive Software Suite for Particle Beam Simulation — Special Feature — Development of a Simulation Framework for Radiotherapy (Japanese). *Japan Society for Simulation Technology*, 28:2–3, March 2009.

[30] Digital Imaging and Communications in Medicine (DICOM). Online. `http://medical.nema.org/`.