A Development of Lightweight Grid Interface

G Iwai[1], Y Kawai[1], T Sasaki[1] and Y Watase[1]
[1] High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba Japan
E-mail: go.iwai@kek.jp

Abstract. In order to help a rapid development of Grid/Cloud aware applications, we have developed API to abstract the distributed computing infrastructures based on SAGA (A Simple API for Grid Applications). SAGA, which is standardized in the OGF (Open Grid Forum), defines API specifications to access distributed computing infrastructures, such as Grid, Cloud and local computing resources. The Universal Grid API (UGAPI), which is a set of command line interfaces (CLI) and APIs, aims to offer simpler API to combine several SAGA interfaces with richer functionalities. These CLIs of the UGAPI offer typical functionalities required by end users for job management and file access to the different distributed computing infrastructures as well as local computing resources. We have also built a web interface for the particle therapy simulation and demonstrated the large scale calculation using the different infrastructures at the same time. In this paper, we would like to present how the web interface based on UGAPI and SAGA achieve more efficient utilization of computing resources over the different infrastructures with technical details and practical experiences.

1. Introduction

Large scale of computing resource is required for the treatment planning in radiotherapy in order to achieve higher accuracy calculation while simulating various physical interactions between particle beam and materials composed of the human body. The interaction processes of an incident beam are independent of each other that allow parallel calculation with a number of CPUs. In order to gain high statistics calculation, we can divide a simulation job into a lot of smaller jobs to be processed in parallel in the distributed computing infrastructures.

Furthermore, our interest is focused on secure job execution in the globally-distributed computing resources of Grid/Cloud infrastructures and an easy-to-use user interface, particularly for medical physists working at hospitals where network connection is severely limited to the Internet. A web user interface can enables such users to execute specific applications and to manage scattered data. Therefore users only have to transfer necessary data through light traffic in networking even if large amounts of data files are handled on their infrastructures.

1.1. SAGA – A Simple API for Grid Applications

As known well, Grid technology has been matured in recent years. However it is not yet enough to say that Grid is utilized as a fundamental infrastructure and is far behind of the primary idea analogized with electrical power Grid [1]. The one of potential reasons for this is that users have to be aware the underlying middleware layer and consequently they also have to make their applications executable in the middleware infrastructures that they are using. It is, therefore, a key subject to provide a uniform architecture to application developers and to offer high-level abstraction layer as a bridge between middleware and application.

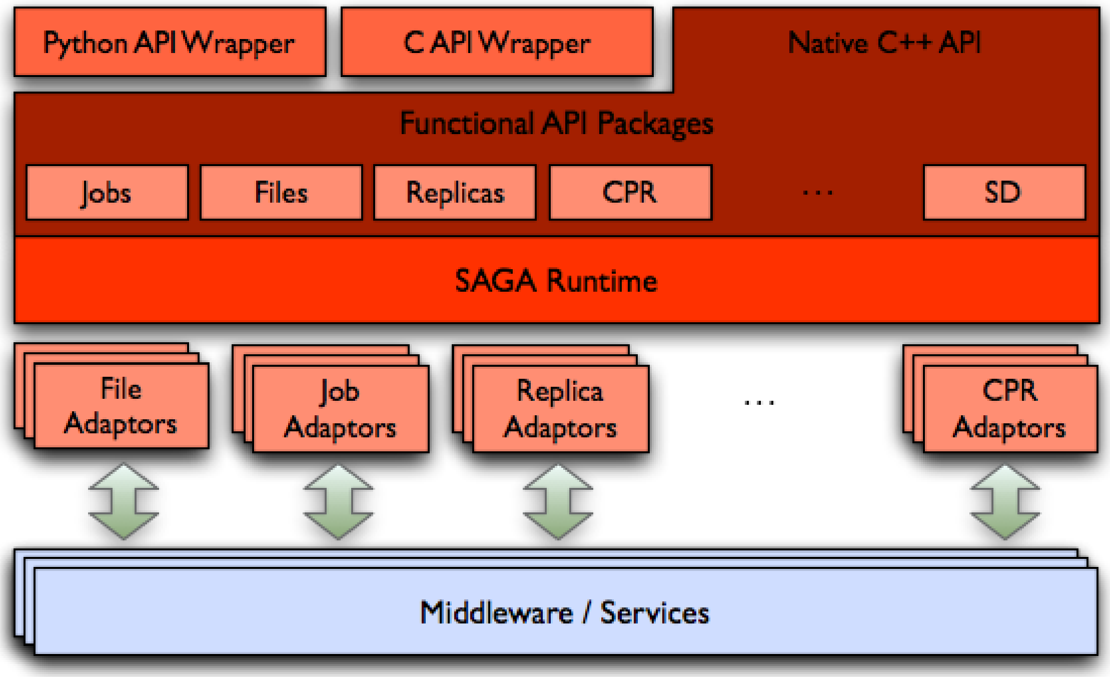


Figure 1. The design of SAGA implemented in C++. (`http://saga.cct.lsu.edu/`)

SAGA [2, 3], which is Open Grid Forum (OGF [4]) standard compliant software, is one of realistic approaches to develop middleware-independent platform in such a complicated situation. SAGA provides a high-level programming interface to Grid functionalities, e.g. job management, file manipulation, to bridge between middleware infrastructures as well as local schedulers such as PBS, LSF, Torque, and so on. For more detailed design of SAGA, it can be divided into two parts of the API and the adaptor as shown in Figure 1. This flexible expandability enables users to submit jobs even in middleware-independently if the corresponding adaptor was implemented.

1.2. RENKEI – Resources Linkage for e-Science
RENKEI project [5] has been launched in 2008 to research this subject with various perspectives. RENKEI is the products name developed in this project and is a general purpose e-Infrastructure composed of NAREGI middleware [6, 7] in Japan. RENKEI aims to develop seamless linkage of resources in Grid, Cloud, and local ones for e-Science. Table 1 summarizes RENKEI collaboration and purposes of each subgroups.

In RENKEI project, KEK aims to demonstrate that SAGA-based applications and platform can work even in multi-Grid environment for end users as well as application developers. In order to realize these, we have developed adaptors, which are required for our research, i.e. NAREGI, PBS, Torque, and so on. In addition to develop these adaptors, We are now developing other adaptors, e.g. Gfarm v1, v2, and iRODS, for different file management system.

UGAPI is an integrated product based on common HEP libraries, SAGA libraries and others as necessary. We have developed a UGAPI-based web user interface which makes distributed-resources interoperable and enables jobs to be executed on multi-Grid middleware infrastructures.

Table 1. Summary of RENKEI collaboration and purposes of subgroups.

Institute	Purpose
Tamagawa University Fujitsu Limited	Develops work flow system to realize middleware-transparent job submitting in national infrastructure as well as laboratory level systems. Also develops application hosting service across infrastructures.
Osaka University University of Tsukuba	Develops distributed file systems to realize efficient access to files from anywhere. Implements RNS based on OGF standard [8], also interoperation with other existing catalogue services.
AIST	Develops a uniform and robust interface to different databases. Also develops user management system and authentication upon products of the interface above.
KEK	Provides framework to develop Grid-enabled application even in multi-Grid middleware and also local resources.
TIT	Dedicated unit for software evaluation in actual resources.

2. Use Cases in Particle Therapy Simulation

Geant4 [9–11] is a toolkit for such simulation and 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 system including graphical interface in collaboration with several medical particle therapy centers. The detail simulation for human body requires many CPU time. It has been obviously represented in previous study that parallel processing is useful to gain the result with higher statistics. Collected use cases in particle therapy simulation related on Grid are listed below:

Distributed job execution
In order to gain high statistics calculation, it is effective to divide a simulation job into a lot of smaller jobs if reasonable clients can be applicable.

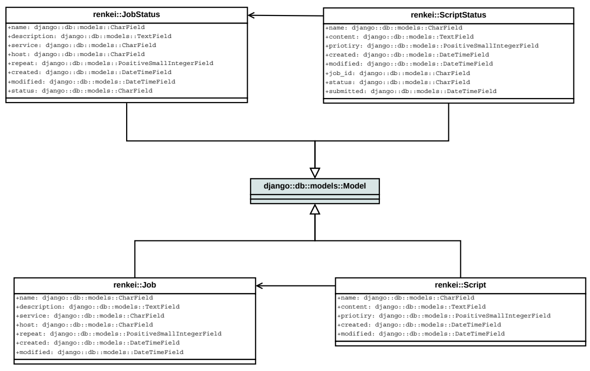
Secure job execution
Secure has two kinds of context in this case. First, users who are working in hospital are typically restricted to access the Internet. Second, patient's personal information must keep inaccessible to unauthorized personnel.

Spatial requirement and storage capability
Large scale data such as DICOM, which stands for Digital Imaging and COmmunication in Medicine and is a standard format for the patient data scanned by medical imaging system, taken from CT need to be stored inside of hospital for the treatment planning.

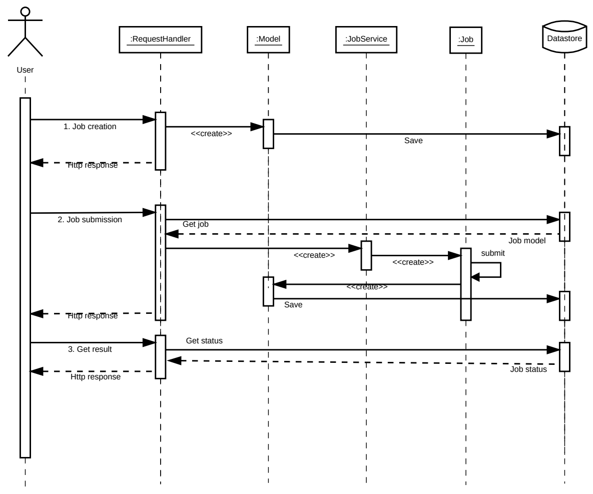
3. Web User Interface

A web user interface has been developed in 100% pure *Python* language. We have a couple of candidates, e.g. Django, Google App Engine [12], CherryPy [13], and so on, for WSGI-compliant [14] web application framework. We are currently using a Django in this prototype by a couple of reasons. The biggest reason for this technical choice is that its high portability and extendability. In addition, since current design of the web interface treats much amount of data in the server host, only Django meets this critical requirement.

Web user interface contains a couple of data models as shown in Figure 2. A data model for Job consists of some fields for its name, its description, service type, execution host, and one or more Script


renkei::JobStatus
+name: django::db::models::CharField
+description: django::db::models::TextField
+service: django::db::models::CharField
+host: django::db::models::CharField
+repeat: django::db::models::PositiveSmallIntegerField
+created: django::db::models::DateTimeField
+modified: django::db::models::DateTimeField
+status: django::db::models::CharField
renkei::ScriptStatus
+name: django::db::models::CharField
+content: django::db::models::TextField
+priotiry: django::db::models::PositiveSmallIntegerField
+created: django::db::models::DateTimeField
+modified: django::db::models::DateTimeField
+job_id: django::db::models::CharField
+status: django::db::models::CharField
+submitted: django::db::models::DateTimeField
django::db::models::Model
renkei::Job
+name: django::db::models::CharField
+description: django::db::models::TextField
+service: django::db::models::CharField
+host: django::db::models::CharField
+repeat: django::db::models::PositiveSmallIntegerField
+created: django::db::models::DateTimeField
+modified: django::db::models::DateTimeField
renkei::Script
+name: django::db::models::CharField
+content: django::db::models::TextField
+priotiry: django::db::models::PositiveSmallIntegerField
+created: django::db::models::DateTimeField
+modified: django::db::models::DateTimeField

(a) A class diagram of data model.



(b) A sequence diagram of web interface.

Figure 2. A design of web interface (essential part only).

data model(s), which has attributes for name, content, and priority. These two kinds of models are defined by users. On the other hands, similar models, –JobStatus and ScriptStatus– are dynamically created upon submitting a job and keep to separate with volatile models, i.e. Job and Script.

Sequence flow among objects and a user is illustrated in Figure 2(b). Users can do actions via web interface by following procedures:

(i) To create a job:
  (a) Users edit a their job and one or more associated scripts on the web form. Then they send a request to the RequestHandler.
  (b) RequestHandler received it and create a model object.
  (c) A model object is saved in the datastore.
  (d) HTTP response is send back to users.

(ii) To submit a job:
  (a) Users submit preferable jobs listed in web browser and send a request to the RequestHandler.
  (b) RequestHandler received it and then retrieve a job model from the datastore.
  (c) A model object is re-created.
  (d) A job service object is instantiated from model object for Job.
  (e) A job object is created by job service and submitted.
  (f) New model object to hold JobStatus and ScriptStatus is created and saved.
  (g) HTTP response is send back to users.

(iii) To obtain results:
  (a) Users send a request to list job status.
  (b) Request handler received it and create a model object for JobStatus and ScriptStatus.
  (c) Request handler generates a list of job results.
  (d) HTTP response is send back to users.

4. Results and Discussion

We have demonstrated a lightweight web user interface of the distributed computing for the proton therapy simulation with the resources of Grid infrastructures as well as local resources. Typical workflow

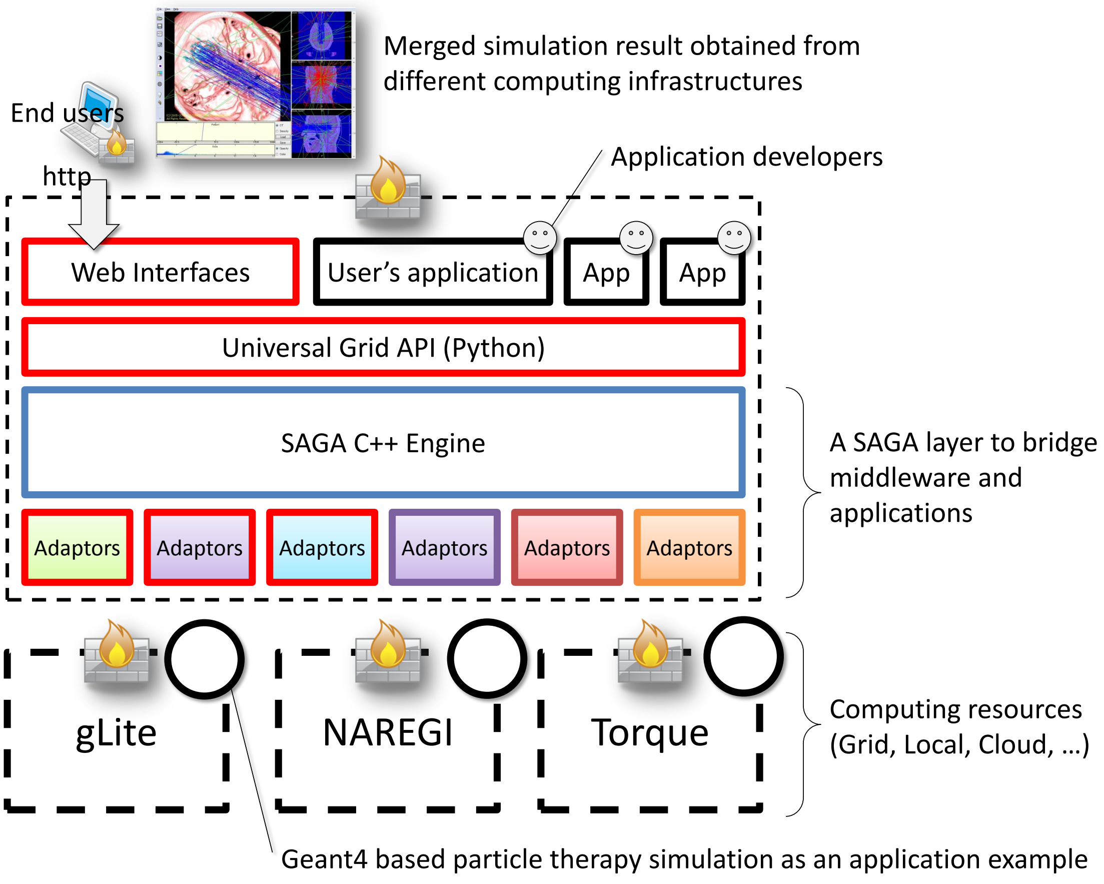

Merged simulation result obtained from different computing infrastructures
End users
http
Application developers
Web Interfaces
User's application
App
App
Universal Grid API (Python)
SAGA C++ Engine
Adaptors
Adaptors
Adaptors
Adaptors
Adaptors
Adaptors
A SAGA layer to bridge middleware and applications
gLite
NAREGI
Torque
Computing resources (Grid, Local, Cloud, …)
Geant4 based particle therapy simulation as an application example

Figure 3. A job execution workflow for the proton therapy simulation through the web user interface.

using this interface is illustrated on Figure 3. For more detail, step-by-step procedures are expressed below:

(i) The Geant4, which provides class libraries for the simulation of the passage of particles through matter, has been installed in the VO specific shared storage space of each calculation machines. The geometrical and material composition data of a gantry and the binary of the simulation program are also pre-installed in the shared space above.

(ii) The simulation script and its input files formatted DICOM are transferred from the Grid storage and staged-in the local file system on calculation machines at the job execution earlier.

(iii) A scripted job is submitted with different seed of random number generator. This work flow can be controlled through the web interface in its creation, execution, cancellation, and so on.

(iv) A job is submitted into the free resources repeatedly until good enough statistics.

(v) After all jobs finish, the output graphical data being stored in a specific storage space and can be downloaded into users client host over the web interface.

(vi) A 3D graphical routine is then initiated locally through web browser to show the particle tracks and dose distribution being super imposed on the DICOM image. This is essential for a medical doctor to evaluate the treatment plan.

We have divided a big job into ∼1,000 jobs in CPUs. We have achieved good enough advantage of scale over the distributed resources. We have produced a simulation result more than 200 times faster than single processing in this example.

5. Future Work and Conclusion

We have developed a UGAPI-based web user interface for particle therapy simulation and have been built on 100% pure *Python*. UGAPI is implemented based on SAGA and provides supplemental and extended functionalities, e.g. legacy HEP functions, that are not shown in SAGA. UGAPI provides API for applications and command line interface based on itself.

This prototyping of web interface have made it possible for users to request most of their operation through it. We have also established to submit same scripted job into NAREGI-deployed resources (RENKEI), local schedulers such as Torque, PBS, and user's desktop machine though uniform sequence. This will enable users to confirm running their own jobs correctly before globally job execution. It often takes much time to fix jobs executed over distributed computing environment, so that users would take an advantage on this uniform platform.

In this demonstration, we have experienced usability of the universal Grid interoperable environment. This also make it possible that non-Grid applications that have ever worked on local resources are portably exported to distributed resources over the Grid resources.

For allowing to use proprietary functionalities in middleware, SAGA certainly requires middleware providers to implement its adaptor if it is not implemented yet. However, it has been confirmed that there is no needs to change in applications themselves even if unknown middleware is newly plugged in their environment.

As the next step for more sophisticated application, we will integrate RNS that is namespace service implementation for based on OGF standard specification. RNS can manage geographically distributed files over the Grid infrastructures and also provides effective file sharing with scale merit. Due to missing implementation in SAGA, we statically allocate resource information for the moment. We would dynamically allocate it and cross-middleware matchmaking after release of Service Discovery packages.

We believe that these can gratefully improve usability of interfaces to the e-Infrastructure toward success of the e-Science and particle therapy simulation.

Acknowledgment

The support of the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) for this work is gratefully acknowledged. The authors would like to thank to SAGA developer team lead by Shantenu Jha of Louisiana State University.

References

[1] Foster I and Kesselman C (eds) 2003 *The Grid: Blueprint for a New Computing Infrastructure* (Morgan Kaufmann Publishers)
[2] Jha S, Kaiser H, El Khamra Y and Weidner O 2007 *Accepted for 3rd IEEE Conference on eScience2007 and Grid Computing, Bangalore, India.* `http://saga.cct.lsu.edu/publications/papers/confpapers/Design-and-Implementation-of-Network-Performance-Aware`
[3] SAGA – A Simple API for Grid Applications Online `http://saga.cct.lsu.edu/`
[4] OGF – Open Grid Forum Online `http://www.ogf.org/`
[5] RENKEI – REsources liNKage for E-scIence Online `http://www.e-sciren.org/index-e.html`
[6] Matsuoka S, Shinjo S, Aoyagi M, Sekiguchi S, Usami H and Miura K 2005 *Proceedings of the IEEE* **93** 522–533
[7] NAREGI – NAtional REsearch Grid Initiative Online `http://www.naregi.org/index_e.html`
[8] Pereira M, Tatebe O, Luan L and Anderson T 2007 RNS Specification (GFD-R-P.101) Tech. rep. GFS-WG `http://www.ogf.org/documents/GFD.101.pdf`
[9] Geant4 Online `http://geant4.cern.ch/`
[10] Allison J *et al.* 2006 *IEEE Trans. Nucl. Sci.* **53** 270–278
[11] Agostinelli S *et al.* 2003 *Nucl. Instrum. Meth.* **A506** 250–303
[12] Google Google App Engine Online `http://code.google.com/intl/en/appengine/`
[13] CherryPy Online `http://www.cherrypy.org/`
[14] Eby P J WSGI Online `http://www.python.org/dev/peps/pep-0333/`