International Conference on Computational Science, ICCS 2010

SAGA-based user environment for distributed computing resources: A universal Grid solution over multi-middleware infrastructures

G. Iwai∗, Y. Kawai, T. Sasaki, Y. Watase

High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801 Japan

Abstract

This paper demonstrates practical applications based on SAGA –A Simple API for Grid Applications– for distributed computing resources over multi-middleware infrastructures. SAGA provides a high-level programming interface that bridges between applications and Grids as well as local schedulers such as PBS.

At the Computing Research Center of KEK, we are playing a role to support not only on-site users, but also domestic university groups in the High Energy and Nuclear Physics (HENP) community. In order to provide a more effective and practical client environment to users, we have developed Grid-adaptive applications based on SAGA as a part of activity in the RESources lINKage for E-science (RENKEI) for the general purpose e-Infrastructure using National Research Grid Initiative (NAREGI) middleware. We present the technical details for the user environment demonstrator and discuss the usability by real HENP applications.

Keywords: Multi-middleware interface; SAGA

1. Introduction

In various emerging scientific activities, scientists need to share knowledge, data, software as well as computing resources. Those who are geographically distributed require uniform access to distributed resources such as provided by a Grid without any specific knowledge of each installation.

The efforts of middleware providers have resulted in more mature Grid technology in recent years. It is, however, not yet widely deployed and utilized as a fundamental infrastructure of research that is far behind from the primary idea analogized with electrical power Grid [1]. One of the potential reasons for this is that users have to be aware of 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 essential to provide a uniform architecture to application developers and to offer a high-level abstraction layer as a bridge between middleware and application.

The RENKEI project [2] was launched in October 2008 to research this subject from various perspectives. RENKEI is also the name of a general e-infrastructure product that is deployed on NAREGI [3, 4] basis. RENKEI
REsources lINKage for E-sclence (RENKEI)

(a) Overview of the relationship between subgroups 1–5 in the RENKEI.

(b) Architecture of the Universal Grid API developed by subgroup 4 in the RENKEI.

Figure 1: RENKEI collaboration diagram and relationship among subgroups are shown on the map. KEK as subgroup 4 is developing Universal Grid API (UGAPI) and SAGA adaptors to integrate legacy HENP libraries and utilize existing resources. KEK also demonstrates application examples based on UGAPI to the HENP communities.

aims to develop the seamless linkage of both local and Grid resources for e-Science. RENKEI consists of members from the Advanced Industrial Science and Technology (AIST), Fujitsu Limited, National Institute of Informatics, Osaka University, Tamagawa University, Tokyo Institute of Technology (TIT), University of Tsukuba, and the High Energy Accelerator Research Organization (KEK). This project consists of five subgroups shown in Figure 1(a) [5] and the institutes involved in each subgroups and their main purposes are summarized in Table 1.

In the Japanese HENP community, KEK is responsible for the coordination and provision of the computing infrastructure, e.g. central network service, software services and so on. KEK is involved in RENKEI in order to utilize efficiently both Grid and non-Grid resources particularly for HENP users. In this project, we will demonstrate that SAGA-based applications can work well even in a multi-Grid environment for end users in this community such as Belle [7], ILC [8], and medical physics.

SAGA [9, 10], is the Open Grid Forum (OGF [11]) standard compliant software and is one of the realistic approaches to realize such an environment independent of the evolution of the middleware.

Table 1: Summary of member institutes and the purpose of subgroups.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Institute</th>
</tr>
</thead>
</table>
| SG1     | Tamagawa University
         | Fujitsu Limited |
| SG2     | Osaka University
         | University of Tsukuba |
| SG3     | AIST      |
| SG4     | KEK       |
| SG5     | TIT       |
In the e-Science community, available Grid middleware environments depend on communities, regions and countries. In order to overcome such differences, we will demonstrate SAGA usability by deploying our application examples in available middleware infrastructures.

Currently at KEK, we operate a number of computing resources with different middleware in order to ensure backward compatibility for users whose applications are specific to their own Grid environments. Table 2 summarizes the current situation where middleware is being used, planned or developed in VOs supported at KEK. As shown in Table 2, some VOs use non-interoperable middleware in their resources. As mentioned, it is a primary responsibility for us to demonstrate a common method to develop and execute an application for at least users who are categorized as application developers in domestic communities. Hence, we will develop an Universal Grid API (UGAPI) shown in Figure 1(b), which is an integrated product based on common HEP libraries, SAGA libraries and others as necessary. In addition, we will provide services and application examples based on UGAPI that make a platform easier to work with without specific knowledge and installation of a Grid.

2. The setup for our demonstration

There are several middleware infrastructures today. Traditionally, using a middleware requires using its own specific commands and rules to execute jobs and to define job descriptions. Such differences embarrass application developers. One of our motivations is to ease their embarrassment and improve their productivity. As mentioned above, SAGA can absorb the differences between different middleware infrastructures. This section describes two types of middleware. One is NAREGI and the other is Torque which is an open source resource manager based on the original PBS scheduler. Our demonstration shows how easy it is to deploy our application examples in different types of middleware. The following is the case of software development using different middleware infrastructures, comparing the SAGA environment with the traditional approach.

2.1. Traditional Approach

In the traditional approach, application developers need to follow the specific commands and rules of each middleware. NAREGI requires its specific command, “naregi-job-submit”, to submit a job with a Work Flow Markup Language (WFML [12]) file. The WFML file is required to define the job description as shown in Figure 2. The other middleware, Torque requires its specific command, “qsub”, to submit a job with a PBS script file as shown in Figure 2. The PBS script is required to define the job description like WFML on NAREGI.

The traditional approach forces application developers to prepare different formats of job descriptions and to use different commands on each middleware, even if the content of job descriptions is same. The application developer should always keep the compatibilities with all middleware that they are using. Also, further efforts are required when user’s applications are deployed in other middleware infrastructures.

2.2. SAGA environment

The abstraction layer provided by SAGA is located as a bridge among middleware infrastructures as shown in the lower right of Figure 1(b) [13].

Once a SAGA adaptor for each middleware is prepared, application developers only need to use the functional API and do not need to care about each specific middleware’s features. Python, C++ and Java API are currently available
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <JobDescription>
        <JobIdentification>
            <JobName>Program</JobName>
        </JobIdentification>
        <Application>
            <POSIXApplication xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix">
                <Executable>test.sh</Executable>
                <WorkingDirectory>workdir</WorkingDirectory>
            </POSIXApplication>
        </Application>
        <Resources>
            <CandidateHosts>
                <HostName>naregi-front.kek.jp</HostName>
            </CandidateHosts>
        </Resources>
    </JobDescription>
</JobDefinition>

#! /bin/csh
#PBS -d workdir
#PBS -q @pbs-server.kek.jp
cd $HOME/workdir
./job_example.sh

Figure 2: An example of WFML and PBS script: NAREGI requires WFML formatted file to be described attributes for a job, e.g. execution file, working directory, and so on. In case of PBS, it is also required to embed some PBS specific instructions in itself.

as the functional SAGA API. Table 3 shows some examples that application developers can invoke APIs to access the middleware.

A SAGA job description has several attributes. The application developer can configure them one time and reuse the job description to submit jobs on other middleware infrastructures. The sample configuration of a SAGA job description will be described in the Section 3.

Application users need only specify the job service, i.e. NAREGI or Torque, and modify a part of job description to switch it to another service.

3. SAGA-based user environment

Figure 3(a) shows a more detailed architecture using SAGA. The SAGA layer is located between “End users” and several kinds of computing resources. Even if a firewall exists between computing resources and higher components,

Table 3: Frequently invoked APIs in SAGA job module.

<table>
<thead>
<tr>
<th>SAGA API</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>saga::url::url()</td>
<td>Specify job service (i.e. NAREGI or Torque).</td>
</tr>
<tr>
<td>saga::job::description::description()</td>
<td>Create a job description.</td>
</tr>
<tr>
<td>saga::job::service::create_job(description)</td>
<td>Create a job with description.</td>
</tr>
<tr>
<td>saga::job::job::run()</td>
<td>Submit a job.</td>
</tr>
</tbody>
</table>
G. Iwai et al. / Procedia Computer Science 1 (2012) 1545–1551
1549

Figure 3: Multi-middleware platform based on SAGA: Application developers can develop their own applications without any concerns about underlying Grid middleware. Furthermore, this approach easily enables to provide a way such as a web interface that end level users could work even behind of firewalls. For the practical experiment, we have deployed a host, which is pre-installed SAGA-adaptors, and a couple of required software libraries.

i.e. SAGA, UGAPI, applications, and end users can use resources from all middleware infrastructures through SAGA adaptors.

KEK, subgroup 4 of the RENKEI, is developing SAGA adaptors for NAREGI (SNA: SAGA NAREGI Adaptor) and for Torque (SPA: SAGA PBS Adaptor) that comply with version 1.0 [14] of the specification discussed in the OGF. The prototypes of both adaptors can be found at the KEK site. Our demonstration works under the SAGA environment with SNA and SPA. The SAGA library, SNA and SPA are installed on a host server that is called “SAGA adaptor host”. All client applications of middleware should be installed on the SAGA adaptor host. In our case, the NAREGI command line interfaces and the Torque client are installed on the SAGA adaptor host. Figure 3(b) shows the workflow diagram for our demonstration.

The same application can be executable in both NAREGI and Torque middleware. Figure 4 shows a sample code to submit 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 the this example (Figure 4), users are just required to specify a pair of job service and hostname as the argument.

For example, a command to submit a job to NAREGI is expressed below:

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

A command to submit a job to Torque is:

$ 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 take care of compatibility between middleware.

4. Results and discussion

Our real HENP applications created in a practical user environment and based upon SAGA has successfully submitted to RENKEI resources on deployed NAREGI as well as local resources managed by Torque. We deployed a Particle Therapy Simulation (PTS) program based on Geant4 [15, 16, 17] as a real application in resources on both middleware. The application is a Monte Carlo simulation of the particle interaction of a proton beam with materials
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 4: Job execution example (sample.py) using python interface.

composed of a human body. The SAGA based python program has successfully controlled the job submission and monitored the job state. The output files of the simulation were transferred to the client host (SAGA adaptor host) and post-processed to display dose distribution and particle trajectories.

The whole process of this workflow was described in a simple python program that is easily readable even by the end users. For application developers, it provides a convenient environment for debugging and tuning the application.

Users can change application parameters or programs even on their front-end hosts (not installed SNA/SPA) and submit jobs to the resources under the both middleware for debugging easily and quickly.

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

5. Future work and conclusion

We have demonstrated that a SAGA-based user environment can co-work in different Grid resources as well as local resources without any concerns about underlied middleware. In order to allow proprietary functionalities in middleware to be used, SAGA certainly requires middleware providers to implement its adaptor if it is not yet implemented. However, it has confirmed that there is no need to change the applications themselves in order to run the application on NAREGII and Torque in this paper. We have also created a common and simple way to execute a job that is based upon HENP applications. Consequently, we have managed to utilize distributed computing resources using practical examples for the midterm milestone in RENKEI.

For the next step in our work, we will integrate HENP libraries and RNS that is namespace service implementation based on the OGF standard specification.

We believe that this can greatly boost the usability of the e-Infrastructure for e-Science.

Acknowledgments

It is a pleasure to acknowledge SAGA developer team lead by Shantenu Jha for their valuable suggestions and support. We also want to thank Adil Hasan for a meticulous revision of the English of the final draft. The support of the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) for this work is gratefully acknowledged.
References