Design, Development and Usage of a Generic Job Submission Grid Service

Enis Afgan, William T. Jones
University of Alabama at Birmingham
1300 University Blvd.
Campbell Hall 115A
++ 1 205 934-2213
{afgane, jonesw}@cis.uab.edu

ABSTRACT
This paper describes the implementation and sample use of a grid service used for general grid job submission required to simplify end-user grid interaction. The user can rely on standard interface and can thus develop necessary clients that would exploit power of the grid without worrying about internal workings and details of the grid. We show a sample client and also give examples of higher level services which can use jobSubmit grid service as a low level tool and extended it to provide a more abstract end-user grid experience, thus attracting more non-expert users to the grid.

Categories and Subject Descriptors
D.2.13 [**Reusable Software**]

General Terms
Design, Reliability

Keywords
Grid service, grid job submission

1. INTRODUCTION
Grid technologies are making great advances in wide-spread acceptance since many initial installation and configuration issues have been resolved, resulting in functional grids across many universities. Still, evident lack of users on those grids is primarily caused by difficulty to develop and/or use already existing applications. The focus of this paper is to describe development, as well as possible uses and extensions, of a grid service used to simplify grid job submission.

Grid services [1], based on Service Oriented Architecture (SOA) [2] and Web Services, can be used as a tool for simplified job submissions as well as client extensions while taking advantage of grid resources. Grid computing not only provides maximum available data/computing resources, but also shows powerful ability on some critical issues, such as security, load balancing and fault tolerance. Grid services provide several key features such as statefull service, notification, and uniform authentication and authorization across different administrative domains. A general job submission service provides all of the above through a well defined interface, and offers a possibility for easy extension. By using jobSubmit service, user can rely on using standard, easy to use java interface and thus develop necessary clients that would exploit power of grid without worrying about internal workings and details of the grid. The effort required for writing a client for jobSubmit service is similar to creating a custom client for a single resource, but by using jobSubmit service's API, the user simultaneously gains access to multiple grid resources. Also, higher level services (e.g. scheduler, portal) can be written to provide grid access at a higher level, abstracting grid middleware details even further from the end user.

ACM SE'06, March 10-12, 2006, Melbourne, Florida, USA Copyright 2006 ACM 1-59593-315-8/06/04....$5.00

2. DESIGN AND IMPLEMENTATION
JobSubmit service is intended to be deployed on each of the resources desired to run on and allow access to house through a well defined set of API. Developing a grid service is coordinated combination of he following five steps:

1. Define the service interface (its operations) in WSDL
2. Implement the interface using Java
3. Define deployment parameters in WSDD
4. Using Ant create a deployment GAR file
5. Deploy the service using a GT4 tool

For more details on individual steps for service implementation, please refer to [3]. The jobSubmit service provides the following API, all of the essential operations involved in submitting a job:

- UploadFile: Upload any files required as input for the job being submitted to the remote resource
- DownloadFile: Download any output files
- jobSubmit: Submit the specified job to the remote resource
- jobStatus: Given jobID, check on the current status
- userNotify: Given jobID, subscribe to the notification mechanism used to automatically report back to the user once the job has completed.

No matter what kind of an application is available on each of the resources, assuming user is able to invoke it, jobSumit service should offer fundamental features to execute it and receive results.

3. USER EXPERIENCE

3.1 Deploying the Service

In order to use the jobSubmit grid service, it is necessary to have a working grid based on Globus Toolkit 4 [4] and running MyProxy [5] server for authentication. Once it is downloaded, the following five steps must be performed on each of the resources:

1. unzip/untar the source code in a desired directory
2. Use globus_build_service.sh to build the necessary GAR file.
3. As user globus (or similar), deploy the service using globus_deploy_gar command.
4. As user globus again, start a globus container
5. Compile/invoke client code and invoke the service

Once the service is deployed, it can be accessed from any remote resource through above given API. It can be used though command line tools or a java client, which is described next.

3.2 Java client

The partial java code showing how to use jobSubmit service is given in Figure 2.

```
1.  public class Client {
2.    public static void main (String[] args) {
3.      JobServiceAddressingLocator loc = new
            JobServiceAddressingLocator();
4.      try {
5.        // Setup service and job requirements
6.        String serviceURI = args[0];
7.        EndpointReferenceType endpoint = new EndpointReferenceType();
8.        endpoint.setAddress(new Address(serviceURI));
9.        JobSubmitPortType job=loc.getJobSubmitPortTypePort(endpoint);
10.       // Setup and submit the job
11.       job.UploadFile ( "titanic", "/home/globus/", "seq.fasta", "everest00",
              "/tmp/", "seq.fasta", 0);
12.       jobID = job.JobSubmit ("everest00", "blastall", "-p blastp -d
              /home/mpiblast/toolkit/yeast.nt -i /tmp/seq.fasta -o /tmp/seq.out");
13.       // Wait for the job to complete
14.       do {
15.       Thread.sleep(1000);
16.       jobStatus = job.jobStatus(jobID);
17.       } while ( !jobStatus.equals("DONE") );
18.       // Get result file
19.       job.DownloadFile ("titanic", "/home/globus/", "seq.out", "everest00",
              "/tmp/", "seq.out", 0);
20.       } catch (Exception ex) {
21.         ex.printStackTrace();
22.       } // end try/catch block   } //end main   } // end Client
```

Figure 2. Client code showing jobSubmit service's API.

In lines 7 and 8, we obtain an endpoint reference of this service which is used to address selected WS-Resource. In the next line we obtain a reference to jobSubmit's service portType which, though the JobSubmitServiceLocator, given service's endpoint, allows us to contact jobSubmit portType. Once we have setup the communication requirements, we are able to use service's interface and perform job submission as if we were working with a local object. As an example of a job, we are submitting a BLAST [6] query to one of the remote resources. First, in line 11 we transfer the query file, seq. fasta. The function requires us to specify path and file name, as well as the resource name where the file can be found and repeat the same for the remote resource. Once the file is file has been copied, we are able to submit the job by specifying resource name, executable name as well as any parameters. In current example, on lines 14 – 17, we keep checking job status until it has completed. Finally, in line 19 we are able to retrieve the result file from the remote resource in a similar way as we transferred the input file.

As is apparent from the given code, using jobSubmit service is about as straightforward as it gets when talking about submitting a job to the grid. The end-user is never really concerned with details of grid job submission. Only requirements are knowledge of selected resource and executable location. As noted before, it is obvious there is a need for a higher level service now which would use these APIs and would coordinate such variables, allowing the user to experience an even more abstract access to the grid. The only aspect missing from the code given is user authentication which must be considered in a real application.

4. CONCLUSION

In the course of the paper, we described the need, the architecture and implementation, as well as shown a simple client showing how to invoke a remote job, and thus, how to use the jobSubmit grid service. The service provides a standardized, high-level set of functions allowing for easy job submissions and access to grid resources. It alleviates the end-user from having to learn some of the cumbersome terminology and details of how to use the grid. There is also a set of tools that could benefit from the above service by relying on its functionality and robustness. At the current level, the user is still communicating directly with individual resource. A higher level service, such as a scheduler or a web portal, could be provided and allow the user to be abstracted even further from grid usage details.

5. REFERENCES

[1] Berman, F., A. Hey, and G. Fox, eds. *Grid Computing: Making The Global Infrastructure a Reality*, New York, John Wiley & Sons, 2003.

[2] Foster, I., Kesselman C., Nick J., and Tuecke, S. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Global Grid Forum, June 22, 2002.

[3] Borja, S. *The Globus Toolkit 4 Programmer's Tutorial*, 2005, [Last accessed 12/14, 2005], Available from http://gdp.globus.org/gt4-tutorial/.

[4] Foster, I. *Globus Toolkit Version 4: Software for Service-Oriented Systems,* IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pages 2-13, 2005.

[5] Novotny, J., Tuecke S., and Welch V., *An Online Credential Repository for the Grid: MyProxy*. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, pages 104-111, 2001.

[6] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. *Basic local alignment search tool*. J. Mol. Biol. 215, pages 403-410, 1990.