Building QuakeSim Portlets with GTLAB

Mehmet A. Nacar[1, 2], Marlon E. Pierce[1], Andrea Donnellan[3], Geoffrey C. Fox[1, 2]

[1]Community Grids Lab, Indiana University
501 N. Morton St. Suite 224 Bloomington, IN 47404 USA

[2]The Department of Computer Science, Indiana University
Lindley Hall Bloomington, IN 47404 USA

[3]Science Division, Jet Propulsion Laboratory
California Institute of Technology Pasadena, CA 91109 USA

{mnacar, marpierc, gcf}@indiana.edu, donnellan@jpl.nasa.gov

Abstract

The QuakeSim portal is a problem solving environment to develop a solid Earth science framework for modelling and understanding earthquakes. In this study, we proposed an evolutionary approach to allow TeraGrid usage in addition to our own clusters for QuakeSim portal. Our approach is based on our Grid Tag Libraries and Beans (GTLAB) libraries, which encapsulate common Grid operations with reusable XML tags. GTLAB enables rapid development of grid portlets rather than typical portlet development techniques. Although it adds a new layer to programming stack, our experiments show that the performance delay is tolerable in Web applications.

Keywords: Grid portals, QuakeSim portlets, Discloc, Simplex

1. Introduction

The QuakeSim portal is a problem solving environment to develop a solid Earth science framework for modelling and understanding earthquake and tectonic processes. The multi-scale nature of earthquakes requires integrating many data types and models to fully simulate and understand the earthquake process. The QuakeSim gateway includes portlets and services for accessing real time and archival data. The data sources (Global Positioning System data, earthquake fault models) can be integrated with computational applications for event detection and seismic deformation calculations. These latter include finite element methods (GeoFEST [1]) that can be computationally intensive and best run on parallelized platforms. In this study, we aim to utilize TeraGrid [2] resources to solve computational problems of QuakeSim project [3].

Grid portals are essential part of science applications on the web. In the last decade, there are many efforts to build Grid computing environments. These are summarized in [4]. Grid portals are gateways to scientific data and applications that facilitate user-friendly interfaces and enable access to data and metadata. TeraGrid is a well known Grid service provider and Virtual Organization[5].

Applications ranging from life sciences to space exploration are accessible by science gateways that serve many education levels in the community. Examples of our work include the QuakeSim [6] portal for earthquake modelling and VLab [7] portal for material science for Earth. In this study, we will focus on QuakeSim portal as specific motivating use case for our work.

Grid portal development depends upon a complicated distributed computing stack. There are several layers of tooling that research groups work on including Grid service implementations, Grid service programming interfaces, Web service interfaces and providers, portlet development tools, portal framework support tools, credential management support tools, and grid account management tools. In this paper we will concentrate on portlet development category. GTLAB is one of our previous efforts to enable rapid development of Grid portals [8].

The QuakeSim portal has served the community since 2002 and is currently undergoing several major revisions. In terms of using the portal frameworks, it initially used the

Jetspeed [9] framework. It has subsequently been updated to use the standard compliant, second generation portal framework GridSphere [10], which is compatible with JSR 168 portlet specification [11]. In its current form, the QuakeSim portal uses portlets developed with the Java Server Faces (JSF) [12] web application development framework. JSF is component and tag based, and allows extensions. QuakeSim portlets are typically designed as clients to remote Web services that constitute the QuakeSim middleware. These portlets aggregate user information and data through JSF interfaces and invoke the actions matching the Web services. QuakeSim services use Apache Ant [13] based services to manage jobs and to build multiple steps of jobs that depend each other (i.e., to handle simple workflows).

QuakeSim's computational services are suitable for many of its applications, but it must be extended to support more extensive computations for parallel applications. Grid services from Globus and Condor provide this capability, so we need a way to modify existing JSF-based portlets to work with these services. TeraGrid is one of the very rich Grid service providers in the North America. They support services ranging from Globus [14] to Condor [15] and Myproxy [16]. To simplify this transition and to provide test cases for our GTLAB framework, we decided to combine the two efforts.

In this paper, we describe the application of GTLAB to QuakeSim as a case study. In this case backend applications run on TeraGrid and we access these legacy applications with GRAM, GridFTP, MyProxy services. We will show the integration and implementation of Disloc and Simplex portlets with GTLAB. We also evaluate development time and runtime performance results based on the tests that we conducted on different geographical locations.

The remainder of the paper is organized as follows. In the next section we give background of QuakeSim components. Section 3 we review Grid tag libraries followed by Section 4 QuakeSim portal architecture. Section 5 reviews QuakeSim portal as case study. In the following sections, we will evaluate the performance and test results then we conclude with Section 7.

2. Background

Disloc [17] is used to calculate surface displacements from earthquake faults. Disloc was used for studying postseismic motions following the 1994 Northridge earthquake, for rapid response to the 2003 San Simeon Earthquake, and for estimating surface deformation from the 2004 Great Sumatran Earthquake. The software produces models of GPS and InSAR data. This program is integrated into the QuakeSim environment as a Web Service invoked by a portlet client. A user can run a model and generate a map of surface displacements. Disloc is useful for assessing possible damage and for comparing against seismologic results and geodetic measurements of surface deformation. It is used for interpretation of interseismic (between earthquakes) strain accumulation, which is important for hazard assessment. It is also used for rapid response following earthquakes. Putting the software into the web-services portal allows for a wider user base, including students and hazard agencies.

Disloc is not computationally demanding but it serves as a prototype for more complex portal applications, particularly GeoFEST mesh generation and computational services. We thus chose this application for our initial test case.

3. Grid Tag Libraries Overview

GTLAB provide a set of JSF tag libraries for Grid portal development. This library encapsulates atomic Grid operations as well as multi-staged operations. We explain GTLAB component model and its job management capabilities in detail as follows.

GTLAB is intended to extend the Grid portlet work of the Open Grid Computing Environments (OGCE) project. As we have discussed previously, GTLAB's goal is to simplify the process for making new Grid portlets. The basic problem is that the portlet component model is too coarse-grained for many science portals and should be supplemented by finer grained components. Individual OGCE portlets encapsulate common Grid functionalities, but they must be adapted by developers to specific applications. Application developers have to customize the portlets to comply with specific needs of the gateway. Another aspect of OGCE is that the capabilities are separated. Developers need to assemble several portlets to get workflow capabilities. All these efforts require substantial effort of programming. The developers need to reuse and modify some of the codes, view pages, configuration and deployment descriptors. However, in some cases the customization is even more complex such as sharing the session memory depends on the Tomcat servlet container. Inter portlet communication is another tricky point in case of trivial portlet applications.

GTLAB attempts to solve these problems by enabling all capabilities within a web application that requires minor customization on the view pages. All other APIs, libraries, and deployment descriptors will be the same. GTLAB architecture provides an abstract and extensible interfaces and APIs. The advantage of this approach is that new tags and beans can be added by deriving the interfaces. For example, Condor and Taverna support can be added in the same way.

GTLAB provides several important features for application developers. First, it provides modular components (tags and beans) to construct science gateway portlet pages. Second, it represents Grid service clients using abstract XML tags. Therefore, portal developers do not need to understand underlying details of Grid services. Finally, it provides a component model for developing Grid portlets out of reusable parts.

Grid users typically must submit jobs to batch queues where the jobs may wait for days or longer before running, and even interactive jobs possibly take a several minutes to finish. Thus we must provide a call-back system that let jobs run while allowing the portal to return control to the user. Thus the GTLAB tags need to track the jobs' lifecycle and monitor their status, displaying this information back to the user. GTLAB creates a handler for every submitted job by the users and displays status information using JSF data tables. These data tables are fed by job handlers that are saved in hash tables within the user session. The visual design of the job monitoring pages is left to application developers so that the developers are able to modify tables and to filter the table values. The users can manage, stop, or cancel running jobs, after they submit them. The job archiving is also tied to job handlers. For example, users can keep good samples, remove old jobs or failed jobs, and otherwise organize their repository. The job's metadata features (submit time, status, finish time, output location and input parameters) are stored and can also be listed.

4. QuakeSim Gateway Architecture

QuakeSim portal architecture was previously designed for Web services invocations in the middleware. These portlets aggregate user information and data through JSF interfaces and the actions invoke matching Web services methods. QuakeSim services utilize Apache Ant-based services for managing executable invocations, interacting with the operating system, and controlling simple workflows. Ant build scripts serve as templates for defining the operations of a particular application service. These server-side Ant build scripts can be converted into portlet-side GTLAB XML tags.

Instead of altering QuakeSim service interfaces synchronizing with Grid services, we remove the Web services layer. Therefore we use Grid services to invoke remote applications, to make file transfers and to provide security. However, we also need to allow implementing workflows within the scope of QuakeSim scripts. In other words, we are able to translate Ant scripts to series of Grid service invocations that are represented as graphs. This new approach has advantages to the previous architecture. First, there is no need to alter service interfaces when the Ant scripts change. Second, in the previous system, service clients cannot access the service layer to change scripts. Therefore the clients have to request required changes that involve additional management efforts as well.

Enabling QuakeSim portlets such as the Disloc interface to work with GTLAB requires a few changes on the portlet pages. First of all we preserve all JSF pages that collect information from users such as input forms and parameters. Next, we replace the JSF form page that invokes QuakeSim Web services with Grid tags. Therefore the embedded Grid tags that are invisible to the end users will call Grid services by using Grid beans. As a result of these simple changes we gain from development time.

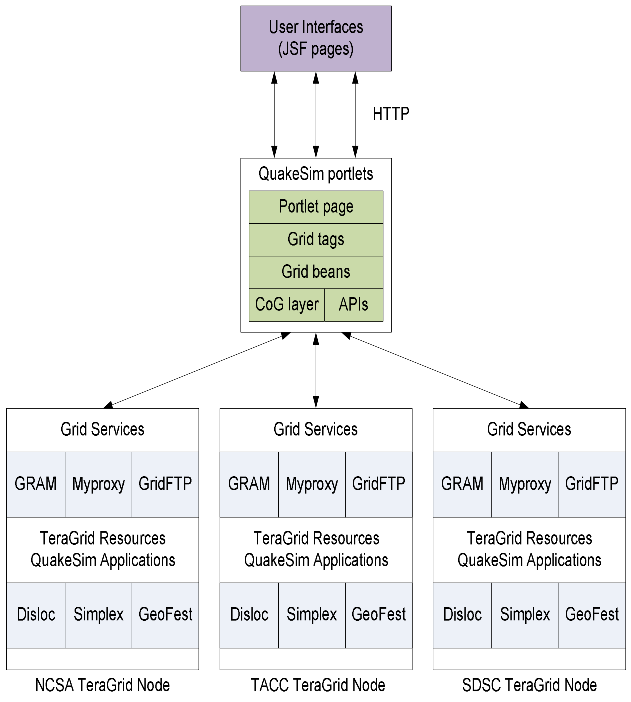

Figure 1 QuakeSim portal architecture with Grid services invocations of TeraGrid nodes.

As shown in the Figure 1, QuakeSim architecture utilizes GTLAB to access TeraGrid nodes. We customize portlet pages to connect which TeraGrid nodes beforehand. Therefore end users would not worry about TeraGrid availability. It is also possible to involve the end users in the node selection stage. In which case, users have to be knowledgeable about the nodes. In our design, users get their Myproxy credentials before using any other Grid service. Then they can use one of the services such as GRAM for invoking applications or GridFTP to transfer files from one server to another.

5. Case Study: QuakeSim Portal

QuakeSim portal is an environment to utilize supercomputers, clusters or even desktops to understand earthquakes. QuakeSim portal previously built and production with different technologies. In this work, we rebuild QuakeSim portal with Grid portlets by integrating GTLAB. Therefore, we choose GridSphere portal framework to build QuakeSim portal. In the building process we provide portlets for QuakeSim applications including Disloc and Simplex.

5.1. Disloc Portlets

Disloc models multiple dipping dislocations (faults) in an elastic half-space. In the view of portlet development, Disloc is an application that we need to run by providing parameters and input files. Disloc run on TeraGrid and the users either can by using command-line tools or shell scripts. But portal users can only access by using Grid services to access TeraGrid in a secure way.

```
<o:multitask id="multi"
  persistent="true"
  taskname="#{resource.taskname}">
    <o:myproxy id="mypr"
      hostname="gf1" lifetime="2"
      password="manacar" port="7512"
      username="manacar"/>
    <o:jobsubmit id="make"
      arguments="/home/manacar/disloc-
      work" executable="/bin/mkdir"
      hostname="gf1.ucs.indiana.edu"
      provider="GT2"
      stdout="/home/manacar/tmp/out-
      make"/>
    <o:jobsubmit id="disloc"
      arguments="/home/gateway/GEMCode
      s/Disloc/input.txt
      /home/manacar/disloc-
      work/disloc.out"
      executable="/home/gateway/GEMCod
      es/Disloc/disloc"
      hostname="gf1.ucs.indiana.edu"
      provider="GT2"
      stdout="/home/manacar/disloc-
      work/out-disloc"/>
    <o:dependency id="dep"
      dependsOn="make" task="disloc"/>
</o:multitask>
```

Figure 2 Disloc portlet page contains multi-staged jobs with DAG representation

GTLAB provides a client layer on top of Grid services that is bridge to the portal users. In other words, the portal users can access Disloc transparently through portal user interfaces (i.e., Web forms). Not only using an application is possible, but also a DAG could run the multiple steps of Disloc such as in Figure 2 making a directory on the file system to save output file, then running Disloc application that depends on the first task.

5.2. Simplex Portlets

Simplex is an inversion code based on Disloc. Similar to Disloc, Simplex applications are run by DAGs that describe order of the tasks and their dependencies. Then JSF pages collect parameters and information about the task to submit it to TeraGrid.

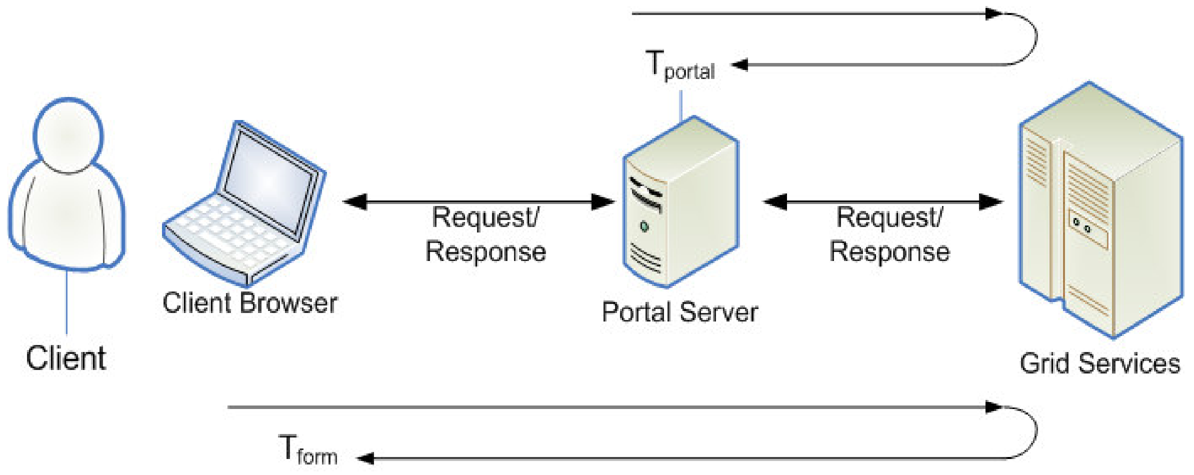

Figure 3 Turnaround times starting from user form and portal server are shown.

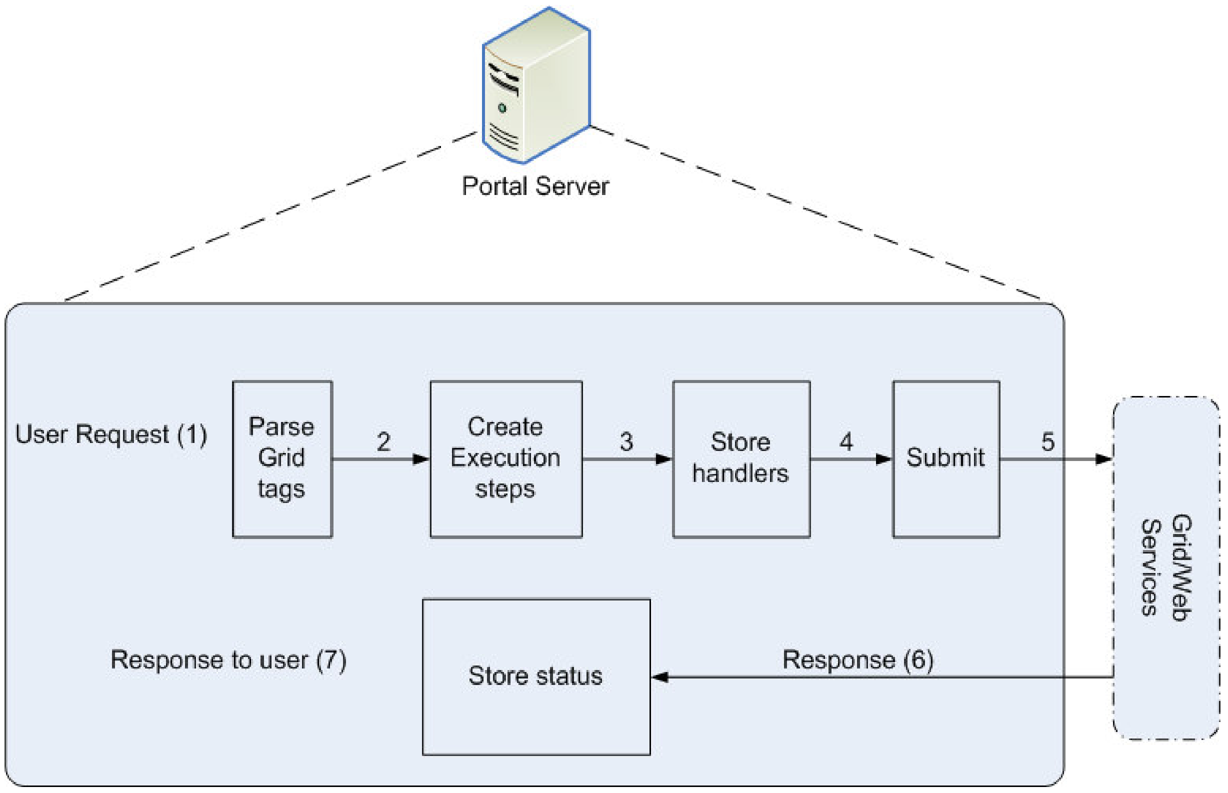

Figure 4 User requests are processed in the Portal server that demonstrated in this picture.

6. Performance Analysis

GTLAB is intended to shorten Grid portal development time, but we must also verify that we do not introduce unacceptable portal performance overhead. We performed run-time tests to analyse GTLAB architecture to determine if it is adding any overhead in the overall processing of the requests. We want to show that the QuakeSim architecture can tolerate the negligible overhead. Our testing baseline and testing framework is explained in great detail in the next section.

The reusing Java bean libraries are the most time consuming task of portlet development. Constructing JSF pages synchronize with the backing beans is another bottleneck. We also automated building of portlets with Maven tool to reduce customization time. While we integrate GTLAB to construct JSF pages that we realized that reduces a time consuming task. We experienced these during the course of several Grid portals including QuakeSim, and VLab.

6.1. Testing Setup

GTLAB testing server runs on Tomcat server and these tests aim to measure turnaround time on the server and client sides. Clients make extensive number of requests to show performance and thresholds. We have measured elapsed time at the starting and at the ending of requests in Figure 3. There are two testing cases: i) Turnaround time for requests are initiated from browser client by submitting web forms denoted T_{form}. ii) Turnaround time for requests from portal server denoted T_{portal}.

The client requests are conducted by using HttpClient programming interfaces [18]. HttpClient provides interface to feed web form parameters and submit buttons. We have embedded JSF pages to make calls to GRAM service to execute a simple Directed Acyclic Graph (DAG). This example DAG first obtains the Myproxy credential from Myproxy repository and then submits simple UNIX commands. We note we are not interested in testing the backend Grid service performance. In order to get elapsed time accurately, we have taken "submitted" message into account as response from GRAM service. Otherwise, the elapsed time may indicate longer time to wait to job is completed because GRAM job submissions are subject to be queued on the GRAM service.

Figure 4 shows detailed scope of processes when request comes to the portal server. At the first stage, user requests (1) are parsed as using JSF component model. The Grid tag components embedded into JSF pages are extracted and then the graph structure is constructed by preserving the dependencies. Grid operations on the graph are converted to Java CoG [19] stubs "taskgraph" in (2). While CoG stubs are created, the handlers of the tasks are stored in session memory (3). Then the submit action is processed (4) that invokes a Grid service. The submission status is stored, as far as Grid service sends a response message it may be a "submitted" which is for success or "failed" which is for failure. Finally, the response is directed to the user session on the Web browser (7).

6.2. Run Time Evaluation

The test results have shown that GTLAB framework has negligible overhead. We have applied the test scenarios to get the accurate overhead. We have done tests on TeraGrid nodes including IU, NCSA and TACC. The average overhead is less than 100 msec that is tolerable in compare with 3-4 sec average HTTP request processing time shown in Figure 5.

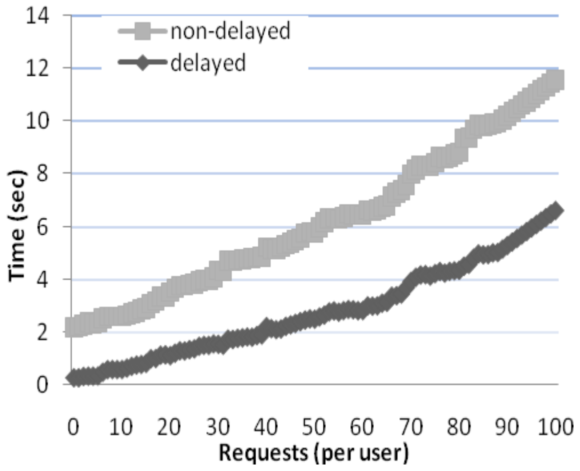

Figure 5 Performance of requests per user

Figure 5 shows the baseline performance results per user. The plots indicate T_{form} turnaround time. We intended to give an idea of how long the requests take to be responded. In this case, there are two plots; one is non-delayed which show simultaneous requests per user. Second, delayed plot shows the requests per user with 10 msec interval time.

7. Conclusion

In this paper, we have reviewed QuakeSim portal architecture that calls Web services in the middleware. We have listed disadvantages of this architecture that is hard to import on TeraGrid nodes. Then we sketched our new architecture to derive Grid services to invoke QuakSim applications like Disloc and Simplex remotely. In that architecture we are not only use Grid services clients, but we also add an additional reusable coding layer that makes Grid service clients portable. As a result, we integrated GTLAB within QuakeSim portlets.

We also analysed the performance of new QuakeSim portal architecture as showing that GTLAB layer add a negligible processing overhead. The GTLAB processing overhead is less than 100 msec in average which is tolerable in case of processing HTTP requests. Therefore we proved that GTLAB does not decrease performance of QuakeSim portal. However it reduces cost of development stage as advantage.

8. References

[1] Parker, J.W., Donnellan, A., Lyzenga, G., Rundle, J.B., and Tullis, T. *Performance Modeling Codes for the QuakeSim Problem Solving Environment.* in *Proceedings of the International Conference on Computational Science (Part III).* 2003: Springer-Verlag, Berlin.

[2] *TeraGrid*. [cited; Available from: http://www.teragrid.org/.

[3] Donnellan, A., et al., *QuakeSim and the Solid Earth Research Virtual Observatory.* Pure and Applied Geophysics, 2006. **163**(11): p. 2263-2279.

[4] F. Berman, G.F., T. Hey, (eds.), *Grid Computing: Making the Global Infrastructure a Reality*. 2003, Chichester, England: John Wiley & Sons.

[5] Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* International Journal of High Performance Computing Applications, 2001. **15**(3): p. 200-222.

[6] *QuakeSim portal*. [cited; Available from: http://quakesim.jpl.nasa.gov/.

[7] Mehmet A. Nacar, Mehmet S.Aktas., Marlon Pierce, Zhenyu Lu and Gordon Erlebacher, Dan Kigelman, Evan F. Bollig, Cesar De Silva, Benny Sowell, and David A. Yuen, *VLab: Collaborative Grid Services and Portals to Support Computational Material Science* Concurrency and Computation: Practice and Experience, 2007. **19**(12): p. 1717-1728.

[8] Mehmet Nacar, Marlon Pierce, Gordon Erlebacher, Geoffrey Fox, *Designing Grid Tag Libraries and Grid Beans*, in *Second International Workshop on Grid Computing Environments GCE06 at SC06*. 2006: Tampa, FL.

[9] *Jetspeed.* [cited; Available from: http://portals.apache.org/jetspeed-1/.

[10] Jason Novotny, Michael Russell, Oliver Wehrens, *GridSphere: a portal framework for building collaborations.* Concurrency - Practice and Experience, 2004. **16**(5): p. 503-513.

[11] Abdelnur, A., Chien, E., and Hepper, S., (eds.) *Portlet Specification 1.0.* 2003 [cited; Available from: http://www.jcp.org/en/jsr/detail?id=168.

[12] Craig McClanahan, E.B., Roger Kitain, *Java Server Faces Specification. Version 1.1.*

[13] *Apache Ant*. [cited; Available from: http://ant.apache.org/.

[14] *Globus toolkit*. [cited; Available from: http://www.globus.org.

[15] Douglas Thain, T.T., and Miron Livny, *Condor and the Grid*, in *Grid Computing: Making The Global Infrastructure a Reality*, A.J.G.H. Fran Berman, Geoffrey Fox, Editor. 2003, John Wiley.

[16] Jason, N., T. Steven, and W. Von. *An Online Credential Repository for the Grid: MyProxy.* in *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10).* 2001.

[17] Okada, Y., *Surface Deformation Due to Shear and Tensile Faults in a Half-Space.* BSSA, 1985. **75**(4): p. 1135-1154.

[18] *HttpClient*. [cited; Available from: http://jakarta.apache.org/httpcomponents/httpclient-3.x.

[19] Gregor von Laszewski, I.F.J.G.P.L., *A Java commodity grid kit.* Concurrency and Computation: Practice and Experience, 2001. **13**(8-9): p. 645-662.