GridSpeed: A Web-based Grid Portal Generation Server

Toyotaro Suzumura
Tokyo Institute of Technology, and
Japan Society for the Promotion of Science
suzumura@is.titech.ac.jp

Hidemoto Nakada
Tokyo Institute of Technology, and
National Institute of Advanced
Industrial Science and Technology
hide-nakada@aist.go.jp

Satoshi Matsuoka
Tokyo Institute of Technology, and
National Institute of Informatics
matsu@is.titech.ac.jp

Henri Casanova
San Diego Supercomputer Center
Dept. of Computer Science and Engineering
University of California, San Diego
casanova@cs.ucsd.edu

Abstract

GridSpeed is a grid portal hosting server that automatically generates and publishes a customized web interface to the grid for applications, with minimal effort required from the user. Users need only to specify information regarding their application using simple GridSpeed web forms. With GridSpeed, users need not make any modifications to their applications nor write any glue code to publish the application on the web, not requiring any knowledge of Perl, JSP (Java Server Pages) or Java Servlets. Moreover, the portal generated by GridSpeed provides an application frontend as well as a set of fundamental portal services such as an information service, monitoring service, data management, single sign-on, and so forth. GridSpeed publishes a set of portals as Grid services themselves generated by the system that is sharable, searchable, and accessible from others interested in using the application. This feature facilitates the reuse of application portals for specific application domains, as well as increases the number of available Grid applications accessible on the web. This paper describes an overview and architecture of the GridSpeed system, and evaluates the system for two real-world scientific applications: BLAST and MCell.

1 Introduction and Motivation

Grid portals have emerged as a high-level tool for application users to take advantage of the Grid infrastructure effectively. While Grid middleware provides a common set of fundamental services and capabilities that are deployed across resources, Grid portals provide a way for application users to access these services transparently via a familiar web interface in a view to executing potentially large-scale applications on Grid resources.

A Grid portal is defined as a web based application server enhanced with necessary software to communicate with Grid services and resources. Consequently, users could access to Grid resources via the portal from a web browser, in a uniform way, and without the need to install any Grid software on their machine. In general, a Grid portal provides application scientists with a customized view of Grid software and hardware resources specific to their particular problem domain and provides a single point of access to Grid resources they have been authorized to use.

According to the survey completed by the Grid Computing Environment Research Group of the Global Grid Forum, current Grid portals can be largely grouped into two categories: *user portals* and *application portals*.

User portals provide a set of fundamental Grid services for a deployed Virtual Organization (VO) [7], including single sign-on, simple job submission and tracking, file management, resource selection, and data management. Today, user portals can be built with the help of "Grid portal construction toolkit" such as GridPort[9] and GPDK[10]. GridPort is probably the most widely used and originally emerged from the HotPage project, which provides users with a view of distributed computing resources, including the status and availability of individual machines. In addition, the HotPage allows users to access files and perform routine computational tasks.

By contrast, application portals are defined as application specific environments for using and programming complex tasks on the Grid, and come in a variety of forms. Some are designed around relatively specific application domains. For example, the Cactus portal from the Albert Einstein Institute was originally designed for black hole simulations. The Lattice portal from Jefferson Labs is an application portal for high energy physics. Currently such application portals have been built from scratch by collaborative efforts between application scientists and Grid portal experts. To date, and unlike for user portals, there are no available toolkits for constructing Grid application portals.

Grid deployments are becoming increasingly mature and several successful VOs have been established for various applications. Current trends show that the number and variety of potential Grid applications will soon increase sharply, and we argue that application portals will be the interface of choice as most scientific users will wish to run applications via the web. It seems infeasible for application portal experts to satisfy all the demand from the new generation of Grid users. There is thus a strong need for tools that can facilitate the development of application portals.

Such tools could come in the form of APIs and development

libraries. In fact, existing user portal construction toolkits are provided as Perl Modules or Java Beans that provide simple interfaces to fundamental Grid services. Typically, the people in charge of building user portals are system administrators managing Grid resources within Virtual Organizations and/or savvy web programmers. Therefore, providing such developers with potentially sophisticated APIs is quite adequate. The same does not hold for application portal developers.

Unlike user portals, application portals are to be built by a much wider spectrum of developers, who are often computer literate but not necessarily trained computer scientists. It is therefore unreasonable to expect these developers to learn web programming techniques and develop their own "glue codes" to publish their applications on the web. Furthermore, application portal developers might need only lightweight portals for validating their applications and conducting temporary experiments for a short period time. Also, an application portal will likely need to evolve at the same pace as the applications and the application's usage. Given these considerations, it is not likely that many application portal developers will consider the investment of learning sophisticated web programming worthwhile.

Consequently, we argue that there is a strong need for an integrated and complete environment that allows application developers to generate their application portals in a straightforward fashion, without having any knowledge of either web programming or Grid programming. In order to achieve this goal, we propose a web-based Grid portal generation server, GridSpeed, which is the focus of this paper.

2 Overview of the GridSpeed System

This section describes the main features provided by GridSpeed.

2.1 Dynamically Generating Application Portals

The key feature of GridSpeed is to allow users to dynamically generate application portals or instantiate/publish them without any programming. This is accomplished by letting users input a set of information about their applications and computing environment in an intuitive fashion through a wizard provided by GridSpeed. This wizard is described in details in Section 4.

2.2 Publishing/Sharing/Reusing Application Portals

GridSpeed provides a repository called the GridSpeed Portal Repository that allows portal creators to publish their generated application portals and application users to search for their target applications while specifying the name, category, manufacturer, physical resources to be used. Figure 1 shows a web interface to the repository.

Generated application portals on the GridSpeed server can be published and shared among other users in a variety of ways. For instance, application service providers can publish their application with a set of dedicated resources, requirement payment for usage on the application on these resources. Some application portal may be published free-of-charge but only for research purposes. Also, GridSpeed provides role-based access control mechanism to limit the usage of registered application portals in the system.

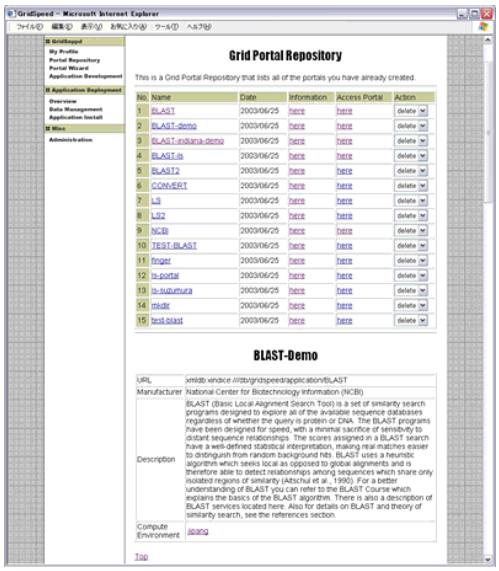

Figure 1. GridSpeed Portal Repository

GridSpeed separates applications from resources clearly. One portal really corresponds to the binding from an application to multiple resources. This separation makes it possible for a portal administrator to reuse an application interface published by someone else, and bind it to his/her own set of resources. This promotes reuse of application interfaces.

2.3 Support For Various Application Structures

For GridSpeed to be useful to a large community of users it must support a wide spectrum of application structures that are relevant to existing scientific applications. In its current version, GridSpeed supports the following three application structure types.

1. a single task to be invoked once,
2. a single task to be invoked multiple times with different input,
3. combination of 1 and 2, with possible dependencies among tasks.

The first type is representative of many applications, for instance ones in which users periodically confront their experimental data to a model. The second type is often labeled "parameter sweep application" [5] and has been shown to be well-suited to Grid executions. Most other scientific applications that have more complex structures are of the third type. GridSpeed offers a simple workflow to describe these application structures.

2.4 Support For Various Grid Middlewares

Different applications may use different computing services such as Globus and SSH, different schedulers such as Condor, Loadleveler, Lsf, PBS, etc., and different data services such as GridFTP and SRB. GridSpeed realizes these requirements by providing a higher-level resource description (bound to tasks) which allows users to specify a large variety of computing services, schedulers, and data services. GridSpeed currently makes use of APST [5] as a workflow engine that handles these resource and

task descriptions, but it is possible to utilize other workflow engines such as DAGMan and Unicore.

2.5 Web-based Single Central Server

The Grid is an inherently distributed environment in which constituent components are managed in a distributed fashion. The current scenario is that, each time a demand arises for setting up an application portal in a VO, the portal administrator launches an application server for that VO. This is not scalable as the demand for application portal increases. In fact, administrating application portals in such an ad-hoc distributed fashion would almost certainly preclude the wide deployment and use of application portals.

Instead, GridSpeed is designed as a logically single server that maintains all application portals in a centralized fashion in order to free portal administrators from deploying application portals on every demand. Note that this does not mean that the GridSpeed server runs applications themselves in a centralized fashion. Rather, applications run on the Grid in a distributed fashion, but the GridSpeed server is to provide user interfaces of application portals in a centralized fashion, not to provide computing resources.

2.6 Grid Application Development and Execution Framework

IDEs (Integrated Development Environments) such as Eclipse, have become increasingly popular tools for building applications. However, these tools aim at providing an application development environment rather than an application execution framework. GridSpeed aims at providing both. Application scientists located in different institutions in a VO often need to work together to build, run, and analyze the results of an application collaboratively. GridSpeed provides an online environment for application developers not only for building applications and application portals, but also for launching and testing the application on their Grid computing environment.

2.7 Generating Unified Fundamental Grid Services

GridSpeed provides a set of unified fundamental Grid services such as single sign-on, job monitoring, file management, resource management, user management, etc. GridSpeed dynamically generates those services for each VO. As mentioned in Section 1, interfaces to the Grid infrastructure are well developed for existing user portals. Consequently, GridSpeed leverages existing user portal development toolkits [9, 10] rather than implementing its Grid services from scratch.

3 Application Scenario - Generating a BLAST Portal

This section describes how GridSpeed can allow application scientists to quickly set up their collaborative environment by applying the system to a biological application, BLAST.

BLAST [4] is a set of similarity search programs designed to explore all of the available biologial sequence databases regardless of whether the query is protein or DNA. The application was developed and is maintained by a group at the NCBI site (National Center for Biotechnology Information). One of the key factors that have popularised the application is the fact that NCBI provides an easy-to-use web interface, that allows one to perform BLAST searches using the format and the parameters of the NCBI BLAST network servers. Recently, there has been increasing demand for setting up portals for running BLAST effectively within virtual organizations, using dedicated computing resources. Furthermore, there is a need for customized interfaces rather than using the fixed interface provided by NCBI. As a result, two organizations including ABCC (Advanced Biomedical Computing Center) and BII (Singapore Bioinformatics Institute) have operated their own customized portal interface for running BLAST on their computing testbeds. These efforts are virtually eliminated by the use of GridSpeed.

For instance, someone at NCBI in charge of developing and providing the BLAST application, might access the GridSpeed web site [2] and publish the application interface by specifying the application structure and parameters in the Grid Portal Generation Wizard (described in detail in Section 4). He might then setup and publish the application portal just as the one officially running at the NCBI site. This is performed by defining the computing environment at NCBI and then binding the application interface to the environment.

Meanwhile, assuming that someone at other organizations such as ABCC and BII need to set up their BLAST portal for their organizations, it is not necessary for them to define the application interface again in the wizard. They can reuse the application interface published by NCBI or can even edit the application interface if necessary. Therefore, the only thing they must do is to install the BLAST application at their site, and define their computing environment in the wizard. Then, they can generate a BLAST portal, by binding the application interface to their own testbed. Another big advantage is that they can make use of the web application server running at the GridSpeed web site, which frees them from launching a web application server at their site.

4 Grid Portal Generation Wizard

We have developed a web-based software called the Grid Portal Generation Wizard. The wizard guides a portal developer through the creation and/or publishing of an application portal with minimum effort, from a web interface. The wizard is organized as a set of web pages in which the user is required to fill in or select answers to questions concerning Grid resources and the target application. In the wizard, an application portal is generated by defining two objects: a computing environment object and an application object, and then those objects are bound together. When defining each object through the wizard, the user is asked whether the object will be either published to the public or restricted to certain groups. Moreover, definitions are independent from each other and thus objects need only to be defined once and can be reused by the same or other portal creators. We illustrate the process of instantiating a portal with the screenshots.

4.1 Defining the Computing Environment

This step is performed by resource providers, allowing them to register their computing environment that can be used by portals generated by GridSpeed. A computing environment is comprised of storage resources (disks) and of compute resources (hosts). Each

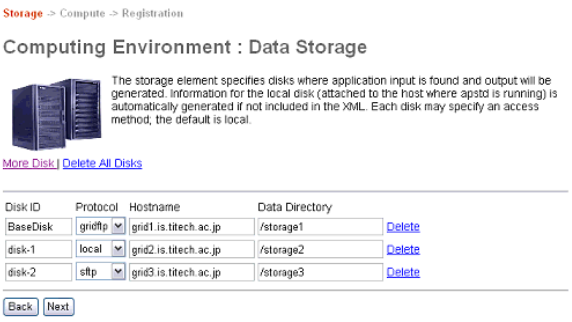

Figure 2. Data Storage

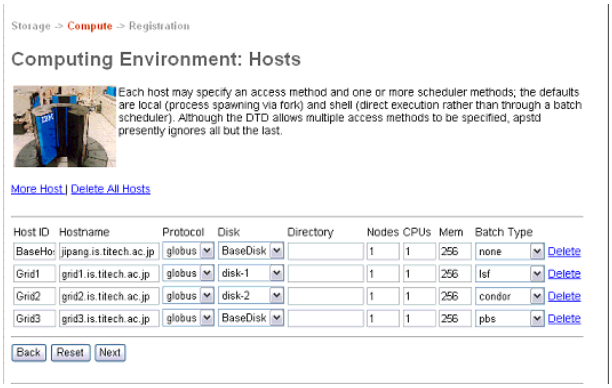

Figure 3. Hosts

disk may specify a host server, a default directory where application input is to be placed and output will be generated, an access protocol such as GridFTP, SRB, SFTP, etc. (See the Figure 2).

The next step (Figure 3) is to define compute hosts. Each host may specify an access protocol such as Globus Gram, SSH, etc, and one or more scheduler methods such as Condor, LSF, PBS, Loadleveler, etc.

4.2 Defining the Application interface

Defining an application interface is performed via the following four web pages: information page, parameter page, template page, and task page, which we describe below.

Information Page This page, shown in the Figure 4, the user fills in a form with various information about the target application (name, subtitle, manufacturer, category, textual description, etc.). Such information is used to categorize applications, allowing one to search for target applications with the keywords. This information is also displayed in the generated application portal.

Parameter Page This page, shown in the Figure 5, the user can decide which parameters should be exposed on the generated portal. This page allows the definition of a parameter by specifying its name, widget type, title, data type, method type, default value, and description. The name is used to be referenced from the Template Page and the Task Page. The widget type is used to represent the actual widget component for the parameter and can be chosen from

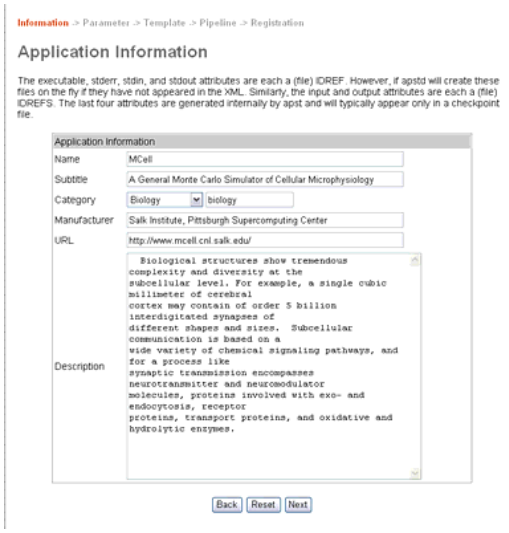

Figure 4. Information

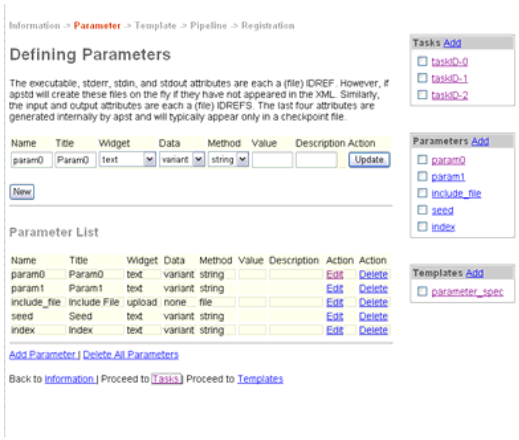

Figure 5. Parameters

”text”, ”textarea”, ”select”, ”upload file”, and ”password”. For a parameter whose widget type is ”select”, one must input a list of items, that is name-value pairs, so that portal users can select one or more of these items. Moreover, a parameter whose widget type is ”upload file”, is the an file uploaded by the portal user. The title is a string placed in front of the widget in which users can input a parameter value. The data type can be chosen from ”integer”, ”double”, ”file”, and ”variant”, and is used to check whether user input data have appropriate data type. The method type can be chosen from ”string” and ”file”. Some parameters must be passed to applications as a file even though they are inputted by users as string. For this case, one can specify ”file” as the method type and then a temporary file containing the value of the parameter is created at runtime and passed to the application.

Task Page In the Task Page, shown in the Figure 6, the portal creator can define an advanced structure for the target application, denoted by “application pipeline” in the wizard. The pipeline is structured as one or more tasks. A task corresponds to the action of launching a certain executable regardless of whether the executable is grid-enabled or not. Each task can be defined by filling in an ex-

Figure 6. Task

ecutable path, input, output, stdout, stin, stdout, estimated runtime, priority, and a description. Dependency between multiple tasks can be controlled by adjusting the task priorities. A task with higher priority can be executed at earlier stage. In each form, it is possible to refer to the actual user input data for each parameter defined in the Parameter Page and the actual instantiated file from a template predefined in the Template Page in the same manner as the Template Page. For instance, the command-line of certain application may look like "-p ${algorithm} -d ${database} -i #{query_file}", where parameters named *algorithm*, *database* must be already defined in the Parameter Page and a template file named *query_file* must be already defined in the Template Page.

4.3 Binding/Publishing

In the GridSpeed context, an application portal is considered to be an interface to the application along with specific set of resources. Therefore, generating a portal is accomplished by binding an application with multiple computing environments through this step. This binding operation realizes on-demand creation of application portals. In the page shown in the Figure 7 the user can generate an application portal by selecting one application and multiple computing environments, both of which must be already defined in Section 4.1 and Section 4.2. After the generation, one can publish the portal to the GridSpeed repository (Figure 1) so that other users can make use of it, or one can add the portal to a list of favorite portals for later usage.

5 GridSpeed Architecture

We have designed and implemented GridSpeed so that it offers the features described in Section 2. The GridSpeed architecture is depicted in Figure 9 and consists of the following components, which we review in detail in the subsequent sections.

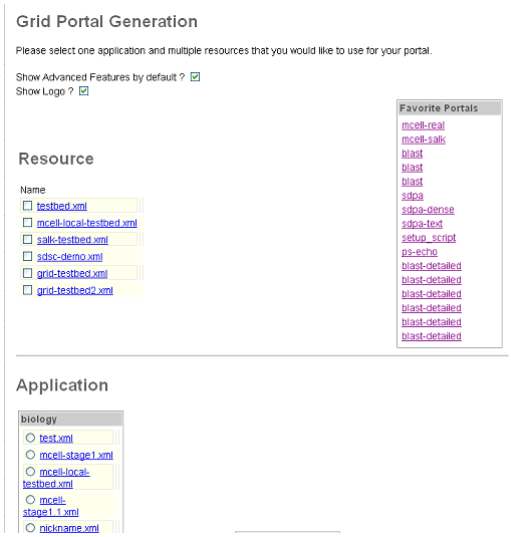

Figure 7. Binding

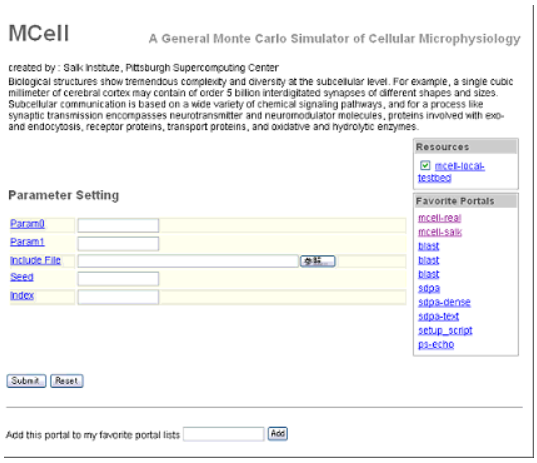

Figure 8. Mcell Portal

5.1 Web Application Server

We utilize the Apache HTTP server as the Web Application Server. The security between the client web browser and the web application server is handled by SSL (i.e. https) using a 56- or 128- bit key. Note that this is one of the primary methods used by commercial portals. For providing web application services, we adopt Jakarta Apache Tomcat which is the servlet container used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.

5.2 Access Controller

Authentication and authorization between the GridSpeed server and the Grid, by default, is handled via the security mechanism of the GSI (Grid Security Infrastructure) protocol. GSI identifies the user's identity via credential signed by the user's certificate. The delegation mechanism realizes single sign-on so that users can access multiple resources on the Grid after they have authenticated

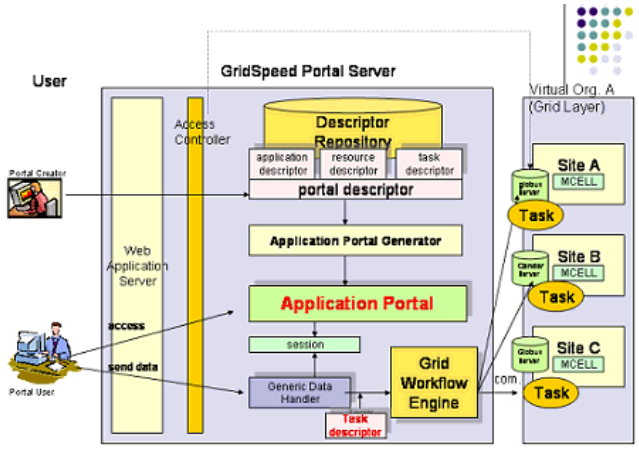

Figure 9. GridSpeed Architecture

to the Grid once. Consequently, if a Grid portal is capable of retrieving a user's credential in a secure way, single sign-on from the portal can be realized.

For this purpose, we make use of an online credentials repository called MyProxy [11], which manages a set of credentials and allows other hosts to retrieve them using username and passphrase. MyProxy has been designed according to strict security principles, such as GSI-protected communications and the non-exposure of private keys. The Java Cog Kit provides a client interface to MyProxy and thus can be easily incorporated into JSP and Servlets with which we have implemented GridSpeed.

5.3 Descriptors

All objects in the system including users, resources, applications, have descriptors that offer their respective detailed information. A user descriptor contains information regarding its account information, a list of generated application portals, and the location of the MyProxy server [11] that is used to retrieve the user's credential. A resource descriptor contains information specified in the phase described in Section 4.1. An application descriptor contains information specified in the phase described in Section 4.2, including application information, parameters, template files, and tasks.

Each descriptor is encoded as an XML document according to an XML Schema defined by GridSpeed. In practice, we employ Castor XML [1] as an XML data binding framework for binding those object descriptors to Java objects model and vice versa. Unlike the two main XML APIs, DOM (Document Object Model) and SAX (Simple API for XML) that mainly deal with the structure of an XML document, Castor makes it possible to deal with the data defined in an XML document via an object model which represents that data. Castor XML can marshal almost any "JavaBeans-like" Java Objects to and from XML.

5.4 Descriptor Repository

The descriptor repository allows for searching, storing, and editing all descriptors. The web interface to the repository is shown in Figure 1. The repository is based on an open-source XML database, Apache Xindice 1.0. Xindice is a "native XML database", meaning that it is designed from the ground up to store XML data. The benefit of a native solution is that one must not be aware of mapping XML to some other data structure. In its current implementation Xindice uses XML:DB XUpdate for its update language and XPath for its query language, to insert data as XML and retrieve it as XML.

5.5 Application Portal Generator

The Application Portal Generator is the core component of the GridSpeed architecture. It generates an application portal interface from a set of required descriptors dynamically loaded from the Descriptor Repository. The generator retrieves required XML documents, which are then marshaled into Java objects via Castor. The generator produces from the objects a JSP file, which implements the actual application portal.

5.6 Application Portal

An application portal is the actual front-end that provides the users with a web interface for filling out input data and specifying computing environments to be used. The application portal, a JSP page, is structured as two separate parts. The first part consists of Java code responsible for storing application and resource description objects into the HTTP session, which are then passed to the Generic Data Handler to specify the way of handling user-submitted data. The second part consists of HTML code that is responsible for handling input. Furthermore, type checking in the input form is handled by JavaScript. For instance, if the user inputs a floating point data into a field declared as integer, a warning would be generated and error detection thus happens at an early stage.

5.7 Generic Data Handler

The generic data handler is a general-purpose component which is responsible for handling data from GridSpeed-generated application portals. By using resource and application description object fetched from the session, the handler first obtains user input data and/or uploaded file for each parameter shown in the application portal. If the method type of the parameter is specified as "file", the handler stores the value into a temporarily created file. If a template file was defined in the Template Page (See the section **??**), the handler then orders the Template Engine (described in the next section) to instantiate an instance file from the template file stored in the application description. Finally, the handler fetches a task description from the application description and then instantiates tasks while applying actual values of each parameter to the task description. The instantiated task object is unmarshaled to an external file in the user's data repository, and then sent to the Grid Workflow Engine.

5.8 Grid Workflow Engine

The Grid Workflow Engine is responsible for submitting a job to the Grid on behalf of portal users. The engine should meet two requirements. First, the engine should provide interfaces to a large variety of Grid systems and provide a single system view of the

Grid. Second, the engine should support workflow management to enable the execution of multiple jobs and also automate the execution of parameter sweep applications.

Currently, we have adopted APST (The AppLeS Parameter Sweep Template) [5] as a Grid workflow engine. APST meets the two requirements describes above: it automates the execution of parameter sweep applications without knowing which resources they are running on and which grid systems they use. APST assigns individual tasks to available machines, copies the input files, runs the tasks, and returns the output files. APST also tries to assign tasks to machines intelligently, using information such as the load and speed of individual machines.

Moreover, APST enables straightforward integration with GridSpeed because all of the objects in the APST system architecture are represented in XML. Another advantage of using APST over related projects such as Nimrod and Condor that targets PSAs, is that APST has a larger emphasis on the efficient co-location of data and computation. Finally, APST differs from these two other systems in that it is designed to target multiple Grid infrastructure environments simultaneously, which is a big advantage for GridSpeed. In fact, APST provides an interface to Condor.

6 Generating Application Portals for Scientific Computing Simulations

In Section 3, we have described the simplicity in generating an application portal for a simple application by the use of BLAST. As mentioned in Section 2.3, GridSpeed has also the ability to deal with more complicated applications such as scientific computing simulations that involve multiple executable stages and parameter sweeping. In this section we demonstrate this capability by applying GridSpeed to a cutting-edge neuroscience application, MCell.

6.1 MCell

MCell [8] is a computational neuro-science application that uses 3-D Monte-Carlo simulation techniques to study molecular biochemical interactions within living cells. MCell is currently being used in over 20 laboratories over the world for practical applications. The generation of a MCell portal is a good test-case for verifying the capabilities of GridSpeed. Indeed, MCell has two important characteristics that many scientific computing simulation tend to hold: (i) it involves multiple stages, each of which involves a parameter sweep (i.e. thousands of independent tasks with different set of parameter spaces) and (ii) it involves an input file masking out a set of parameter variables.

The advantage of generating an MCell portal is twofold. First, it would greatly ease the process of running large-scale distributed MCell simulation on Grid resources. Second, computational neuroscientists often need a remote collaborative environment to conduct their experiment using their proposed computational models. Furthermore, MCell can be applied to a large variety of application areas including computational chemistry and biology. These users must be able to quickly setup their portal for their collaborative experiments using different set of computing resources. GridSpeed makes it possible to instantiate such environments for each such projects on demand.

6.2 Generating a MCell Portal

When dealing with a relatively complex application such as MCell, the most important points to consider upon generating an application portal through the GridSpeed wizard (See Section 4) are: how is the application structured, and which parameters should be preferably exposed on the generated portal. For MCell, the application is basically structured in three steps: preprocessing, parameter sweeps, and postprocessing. We now detail how one can define these three steps to generate the MCell portal.

Preprocessing The preprocessing phase produces a number of MDL (MCell Description Language) files required for parameter sweeps conducted in the second step. This step is handled by a Perl script which takes a configuration file that presents a list of parameters and a range of values for parameter sweeps.

To make these parameters available from the generated MCell portal, the portal creator first goes to the Parameter Page and defines each parameter. Next, the creator defines a template file. This is accomplished in the Template Page[1], using the defined parameters to dynamically instantiate a configuration file passed to the aforementioned Perl script.

Finally, the creator defines a task by specifying the executable path for the Perl script and the argument format in which one can use the template parameter (in the form of # followed by the name of the template file enclosed between parentheses, as in #{parameter_specfile}).

Simulation - Parameter Sweeps The second step performs parameter sweeping, consecutively invoking a MCell executable with each of the MDL files generated in the preprocessing phase. In the wizard the user can create a new task by specifying the executable path of MCell and the argument format as, "seed_${seed}_index_${index}.mdl", where the two parameter named *seed* and *index*, must be already defined as a variant parameter in the Parameter Page. Next, the creator is asked to choose the method to expand variant parameters in the task description. This question comes from an internal expression in a task description represented in the GridSpeed XML Schema. To support a compact description for a number of tasks, GridSpeed allows the incorporation of a variant parameter within a task description. A variant parameter has an attribute called *expand*, for which one can choose from '*string*', '*task*', '*work*', and '*value*', depending on how the task description is to be expanded.

Postprocessing The postprocessing step performs statistical analysis on a set of one or more time-series files. The input files are produced in the previous stage and the argument forward can be specified as "-avg seed_${seed}_index_${index}.output" where both *seed* and *index* are user input data. These two parameter are specified as "variant', and one specifies the expand attribute as "word" in order to expand all input files in line within the argument.

[1]The wizard has an additional page called the Template Page which allows users to define a template file, although there are not enough space for describing the details about it in this paper

7 Discussion

The most daunting conceptual challenge presented by the creation of application portals is the issue of how application-specific interfaces should be created in a generic fashion.

We showed in Section 6 that GridSpeed entails sufficient capabilities to deal with real-world scientific applications. The portals we have generated with GridSpeed have proven very useful for launching and controlling jobs and then obtaining outputs. However, some application scientists need to perform their work while cooperating with external application-specific tools. For instance, MCell users utilizes a Java-based application for visualizing output stored in the database and analyzing the quality of input models in order to decide if further experiments are needed. In other words, they require computational steering. Although such external components varies among applications, it seems that, ultimately, it will be necessary for GridSpeed to have the capability for loading additional user interface components on demand.

To dynamically add on application-specific features, we have been investigating the notion of Portlets which is becoming an increasingly popular concept used to describe a visual user interfaces. Portlets represent modular, reusable software components that may be developed independently of the general portal architecture and offers a specific set of operations. Multiple portlets can be aggregated in single portal page providing users with easy access to a range of services/applications suitable for a particular problem domain. In the portlet framework, GridSpeed would be used to generate a single portlet that is responsible for an application portal, alongside which users could "place" any number of application-specific portlets.

8 Related Work

As mentioned earlier, portal efforts by GridPort [9] and GPDK [10] complement our system. The core implementation of GridSpeed is mainly focused on automatic generation of grid application portals. It does not entail the re-invention of interfaces to core Grid services that have been built by these two projects.

Not many project so far have targeted the generation of application-specific portals. Nevertheless, we can cite NEOS [6], a general client server system that is dedicated to the mathematical optimization area. The key concept is somewhat similar to that of GridSpeed in that the system also allows optimization software developers to easily add-on new solvers to the NEOS system to make those codes remotely available from the web. However, GridSpeed would fit better with the Grid paradigm. NEOS has no capability to spread out the job to multiple resources on distributed infrastructure as the Grid because the communication between client and server is performed through plain sockets and has no grid-compliant authentication and authorization frameworks. Additionally, the generated user interface is not dynamic and it is not straightforward to generate it because the configuration file must be provided by solver administrators when they add their solver to the NEOS Server. The types of applications are limited as well. NEOS only accepts optimization solvers that can be adapted to read input from one or more files and to write meaningful results to a single file. Consequently, user interfaces generated by NEOS are not as flexible as those generated by GridSpeed.

9 Conclusions and Future Work

In this paper we have described the GridSpeed system, which fully automates the generation of grid application portals. When applying GridSpeed to the scientific applications described in Section 3 and Section 6, the GridSpeed system has proven to be flexible and robust enough to allow scientists to make their applications accessible on the web in a straightforward manner. We anticipate that the reduced cost and time associated with creating and deploying these portals with GridSpeed will make portal development easier and more appealing to scientists who which to migrate their applications to the Grid.

For future work, we plan to increase the number and types of applications available from the GridSpeed server. Another future direction would be to enhance the APST system employed as the Grid workflow engine in the current implementation. As stated in the architecture section, an APST daemon is launched for one user because APST aims at the user-level scheduling. Clearly this architecture is not scalable for our purpose in terms of number of simultaneous users (which was not a concern of the APST project initially). Furthermore, there would be possibility to rebuild the current C-based APST daemon in Java, which would allow more seamless integration with JSP/Servlets on the GridSpeed server, as well as allow for stricter security.

At the moment the first prototype of our software is available from the GridSpeed web site [2].

References

[1] Castor. http://castor.exolab.org/.

[2] GridSpeed. http://www.gridspeed.org/.

[3] NCBI BLAST. http://www.ncbi.nih.gov/BLAST/.

[4] H. Casanova and F. Berman. *Grid Computing: Making the Global Infrastructure a Reality*, chapter 33. John Wiley & Sons Publisher, Inc., 2003.

[5] M. M. P. CZYZYK, J. and J. J. MORa. The Neos Server, 1998. http://www-neos.mcs.anl.gov/.

[6] I. Foster, J. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, June 2002. Open Grid Service Infrastructure WG, Global Grid Forum.

[7] E. J.R.Stiles, T.M.Bartol and M.M.Salpeter. Monte carlo simulation of neuromuscular transmitter release using mcell, a general simulator of cellular physiological processes. In *Computational Neuroscience*, pages 279–284, 1998.

[8] S. M. Mary Thomas and J. Boisseau. Development of web toolkits for computational science portals: The npaci hotpage. In *Proceedings of HPDC 9*, pages 308–309, August 2000.

[9] J. Novotny. The Grid Portal Development Kit. In *Concurrency and Computation: Practice and Experience 14*, 2002.

[10] J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy, 2001.