GRID ENVIRONMENT WITH WEB-BASED PORTAL ACCESS FOR AIR QUALITY MODELLING

BARBARA CHAPMAN, HARI DONEPUDI, YUPENG LI, PRIYA RAGHUNATH, YONGHONG YAN, BABU SUNDARAM* AND JIWEN HE†

Abstract. Air Quality Modelling is a recent important application that addresses several modern environmental health concerns. It requires considerable compute power to build, deploy and analyze the results of an air quality model and efforts to develop such applications can benefit from the availability of a computational grid. In this paper we discuss the needs of an air quality modelling project and describe the campus grid environment that we have built to support this effort. The environment includes a portal with a number of features that go beyond the functionality provided by major existing portals.

Key words. Computational grids, Air quality model, Grid portals, Resource management, OGSA, Customizable portals, Globus, Resource policies, Workflow

1. Introduction. Advanced scientific and engineering problems require a large, pervasive and reliable supply of computing resources. Such requirements cannot always be satisfied by a single organization. A Computational Grid is a virtual organization [1] that combines computational resources from different administrative domains. They may be built to meet the needs of a very large application or project, or they may be constructed to improve resource provision to a group of users, often by increasing the quantity and kind of resources available to individuals. Campus grids are such virtual organizations consisting of resources owned by various labs and departments on a university campus that may be collectively put to use. In an effort to provide better services in our necessarily limited budget, as well as to enable computations such as the one described in this paper, we have begun to collaborate with our Computing Center to build such a grid on our campus.

A number of problems must be solved if a variety of resources in different administrative domains are routinely deployed in concert. One of these is the need to enable users to login to all machines as if they formed a single, jointly managed heterogeneous cluster. Another is to provide help for submitting jobs to one or more machines in this grid in a location-transparent manner; this involves being able to transfer files between different storage resources on the grid, and starting jobs on behalf of the user, typically via a local scheduling system. Since data and other code is frequently transferred between different domains, security concerns must also be addressed. Portable grid software [2, 3], so-called *middleware* that provides much of this functionality, is widely available today and is maturing rapidly; its deployment can provide a viable strategy for institutions such as ours to meet service goals via a grid. However, in addition to the administrative and organizational challenges that have to be met – these include such issues as providing system-wide user identification and management – progress must be made in the provision of user-level grid services if grids are to be routinely used. Middleware solves these problems at a relatively low level; yet users demand more transparency in their interactions with "the grid".

Web based portals provide users with a convenient, ubiquitous and intuitive interface that acts as a single point of access (single sign-on) to a grid and its resources. In addition to serving as a login window, a portal may enable the submission and management of scientific applications in a grid. If carefully designed, it can shield users from the heterogeneity of grid hardware and the complexities of the grid software, as well as resource-specific or scheduler-specific configuration details. There are a large number of on-going efforts attempting to

*Department of Computer Science, University of Houston, Houston, TX, 77204-3010 (Email:chapman@cs.uh.edu)

†Department of Mathematics, University of Houston, Houston, TX, 77204-3008

provide a unified, web-based single point of access to a set of grid resources. However, many such efforts are either specific to an application (area) or a grid environment.

Air quality modelling (AQM) [4] attempts to provide precise and reliable air quality forecasts for a given geographical region and time frame. A number of on-going projects are working to develop such models based upon meteorological as well as chemical processes. They have large-scale computational requirements during both development and deployment phases, and may need to be executed within a strict timeframe. In this paper we describe an interdisciplinary AQM project at our university and show how we are working to meet its computational requirements. Our efforts include building a computational grid on our campus in order to harness the available resources for its execution, and developing and deploying a complete customized version of a general-purpose portal that supports a variety of project tasks including project management, job submission and monitoring, and the subsequent viewing of a job's runtime behavior. We have designed the software used to create the portal environment in a manner that enables us to rapidly build custom portals for projects that intend to use the capability of the grid. One of the challenges that this work entails is the need to adapt to evolving standards for the provision of grid services. We also discuss the emerging Open Grid Service Infrastructure (OGSI) standard [5] and indicate how we are attempting to take it into account.

In the following, we first describe the AQM project in Section 2 below, which briefly describes the applications at the heart of this project and the campus grid that we have built to help meet the execution needs. We then discuss portals for accessing grid functionality and state the main requirements of the AQM project in this respect. Afterwards, Section 4 introduces EZ-Grid, our portal system that has been customized to support AQM. We conclude by briefly discussing related work and our plans for the future.

2. Air Quality Modelling. Every day, the average adult breathes over 3,000 gallons of air. Children breathe even more air per pound of body weight and are thus more susceptible to air pollution. Millions of people live in areas where urban smog, very small particles, and toxic pollutants pose serious health concerns. Long-term exposure to air pollution can cause chronic health concerns, such as cancer and damage to the body's immune, neurological, reproductive, and respiratory systems. Air pollution can be wide-ranging as well as persistent. Many air pollutants, such as those that form urban smog and toxic compounds, remain in the environment for long periods of time.

Many on-going efforts aim to provide precise and reliable air quality forecasts for different areas and some [6, 7] have been put into daily public service. Most forecast systems possess similar frameworks, but different modelling components are chosen for basic functionalities. These modelling components are usually compute-intensive numerical programs parallelized via MPI. Scripting languages are generally deployed to integrate and automate the Fortran or C simulation codes running on supercomputers. This approach usually lacks flexibility and impedes system development because the heterogeneity of different execution environments has to be dealt with at the user level.

2.1. Air quality studies at University of Houston. The Air Quality Modelling (AQM) project of the Institute for Multidimensional Air Quality Studies (IMAQS) at the University of Houston analyzes the air quality in the Greater Houston area and works with national, state and local leaders to provide solutions for Houston's air quality problems, which have resulted in several violations of the ozone standard between 1997 and 1999.

One of the tasks of this project is to develop and apply state-of-art air environmental models and provide precise and reliable air quality forecasts for the Greater Houston Area. There are three major components (Figure 2.1) in our forecast system: a numerical weather forecast system, MM5 [8, 9, 10, 11] which may be replaced by a more advanced model WRF [12, 13], an emission inventory processing package, SMOKE [14, 15, 16], and the chemical

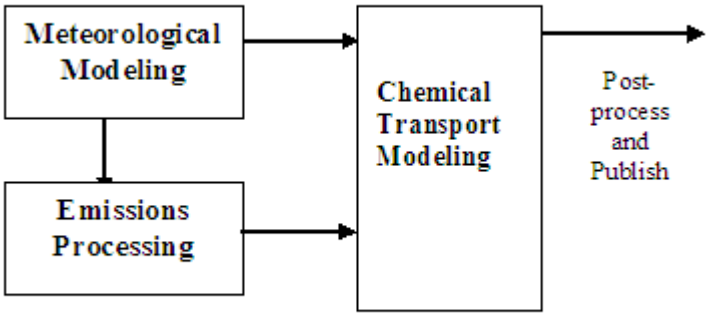

FIG. 2.1. *General application framework for air quality modelling.*

transport model, CAMx [17] or CMAQ [18]. To forecast air quality, several application subsystems, each with multiple processing steps (Figure 2.2), must interact. Large amounts of computing resources are needed on a daily basis. Different parts of the software system must routinely run on different platforms within the campus grid that we describe below. For example, we are currently using the popular weather code MM5 for limited area weather forecasting and reanalysis. In order to prepare the meteorological data as input to the emissions and chemical transportation subsystems, the NCEP input data for MM5 itself must be obtained, preprocessed to create the initial conditions, the simulation performed and post-processing tasks carried out. For a typical setup of the daily forecast of the greater Houston area, MM5 runs on 20 processors of a cluster of Sun SMPs and takes more than 15 hours. The size of the output data produced is more than 3 GB; this data is subsequently transferred to another machine on campus and visualized. Before and after the simulation, several pre- and post-processing steps are performed. The total cost in terms of CPU time as well as storage of results exceeds the amount of available resources at our centrally located HPC Center. Participants come from different research groups and managing the jobs that they individually launch, as well as the output they generate, is a non-trivial task.

2.2. Target environment. Air quality studies including weather and air quality forecasts always demand huge amounts of computing resources. UH has both large and small computational clusters connected through optical fiber across the campus; we exploit their availability by operating a number of different systems, including those at the High Performance Computing Center (HPCC), in a campus-wide grid. The campus grid (Figure 2.3) currently consists of a heterogeneous cluster of Sun SMPs and an SGI visualization system, including a total of 9 TB storage, at the university's HPC Center, along with a cluster of Sun SMPs in Computer Science and several Sun workstations and Linux clusters operated by the Math Department and Geophysics Department. The AQM effort has access to two additional Myrinet-based 64 node Intel clusters and 5 TB storage for further experimentation. This grid is available to a cross-section of faculty and is deployed for both research and teaching. The AQM project and its team members are among the heaviest users of the campus grid.

The campus grid is divided into several different administrative domains corresponding to the owner of the hardware; each domain may contain multiple clusters with a shared file system. In our environment, all basic grid services, such as security and authentication,

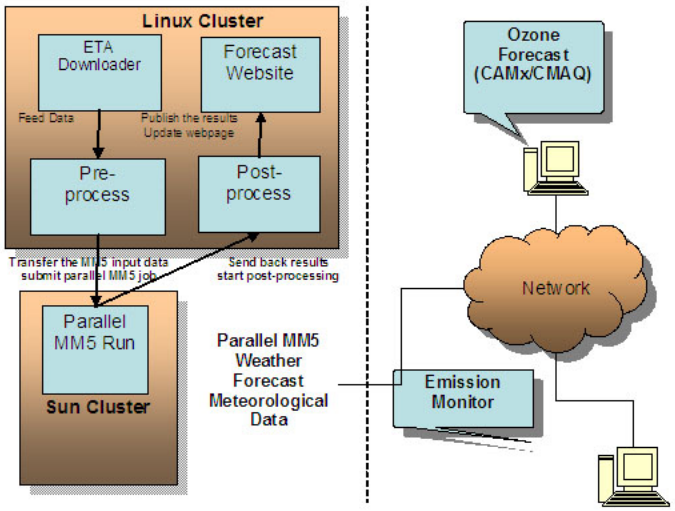

FIG. 2.2. *Real-time MM5 weather forecast at UH*

resource management, static resource information, and data management, are provided by the freely available Globus toolkit, an implementation of a standard set of services for building a grid. Sun Grid Engine (SGE), another free software package, has been installed to manage the resources within the individual administrative domains. We outline their features below. Users may interact with the grid by using Globus and SGE functions directly or they may use our portal (as described in Section 4) to exploit the grid's resources.

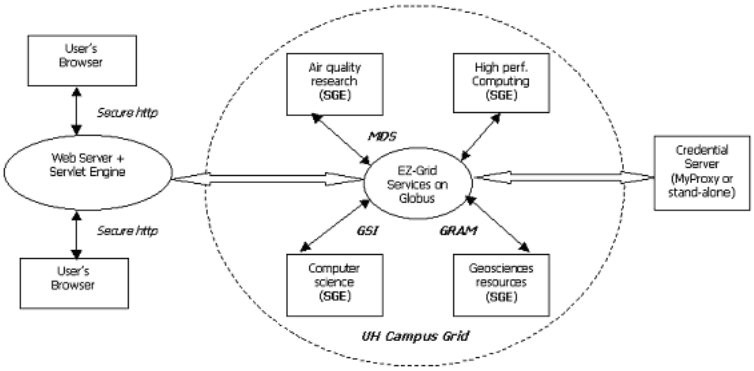

FIG. 2.3. *University of Houston Campus Grid Setup*

2.3. Grid Software, Globus and Sun Grid Engine. The Globus project has pioneered fundamental technologies for building computational grids that harness geographically distributed hardware and software resources. It is widely distributed and is available on a large number of platforms. Thus it forms the basis of many on-going efforts to provide a computational grid, and is considered to be standard infrastructure. Globus middleware is provided as a toolkit of functions that may be used individually or together. They provide the following three major services:

- *resource management* (GRAM) [19], which accepts a job submitted to the grid via a script in Globus' own scripting language, RSL, and passes it on to the resource management system on the selected target for actual queueing and execution, thereby acting as an interface to various kinds of scheduling systems,
- *data management* (GASS, GSI-FTP and GridFTP) [20, 21] that primarily serves to transfer data between different locations on the grid, so that this does not have to be performed by the user before and after jobs are submitted, and
- *information services* (MDS) [22] which provide detailed (static) information on the resources configured in the grid. For compute resources, this comprises hardware details, operating system, other software configurations and more.

These three services are built on top of the underlying Grid Security Infrastructure (GSI) [23], whose functions include single/mutual authentication, encrypted communication and credential delegation. It is essential for security as well as for accounting that a job is associated clearly with the user who submitted it. User access rights and permissions will be used to determine whether and with what priority a job is executed on the machine selected by the user. In the main, PKI (Public Key Infrastructure) has been used for the underlying authentication.

Earlier versions of Globus (GT1,2.x) adopted a toolkit-based approach to realizing grid infrastructure. Globus Toolkit 3 (GT3) is an emerging major revision of this software that aims to provide Globus functionality in the form of web services. It uses technologies such as the Simple Object Access Protocol (SOAP) [24] and Web Services Description Language (WSDL)[25] to enable the creation, management and discovery of grid service instances in a secure and fault-resilient manner. GT3 is based upon Open Grid Services Architecture (OGSA) and related specifications (OGSI), emerging standards which attempt to define what a service must provide if it is to function as a grid service; we describe OGSI in Section 4.4 further below.

Sun Grid Engine (Enterprize Edition) [26] is the local job scheduler and resource manager at each administrative domain of our campus grid. SGE manages heterogeneous collections of machines that share files, possibly via NFS. It provides a variety of mechanisms for specifying domain-specific access priorities and can handle job submissions to an architecture rather than an individual machine. Globus resource management interfaces with SGE as follows. When the user submits a job via RSL, the GRAM component of Globus processes the script and determines the target machine as well as the location of input files: an RSL script will generally specify the kind of job and the name and location of the executable as well as of the input and output files. The local GRAM typically interacts with the GRAM components installed on the corresponding platforms in order to handle data transfers and to dispatch job submission requests to the scheduler managing the selected resources; the scheduler starts the job locally in accordance with the domain-specific scheduling policy, evaluating any arguments passed to it by the job. It also reports resource usage information to the Globus information service.

We have created a job manager to enable interaction between Globus and SGE so that the above functions can be performed. Its task is to do the following: First, it parses the job resource request made via the Globus scripting language (RSL) or via our portal (see below) and uses this to submit job requests to the SGE `qmaster` daemon in the appropriate domains. If necessary, it will send callbacks to client systems. Finally, it will automatically transfer output files to other grid systems using GASS if remote locations have been specified.

With the help of the job manager, different cluster resources can be selected based on their loads and their high-level scheduling and reservation policies. The job manager collects resource usage information from the local resource manager, which records and is able to provide a variety of status information about the cluster.

Certificates form the basis of the authentication process: when a job is submitted to a machine on the grid, the embedded certificate will enable GSI to identify the owner of the job. An independent certification authority (CA) for the campus grid is managed by HPCC. Certificates are issued to the users when their grid accounts are set up. Existing Globus CA or other trusted CA credentials may be exported to the credential repository using secure `http` sessions. Our model for handling credentials is explained further below (Section 4).

For many application scientists, it is a daunting task to deal with a grid infrastructure; we therefore provide a portal interface to simplify interactions with grid services. In the next section, we discuss portal technology and the specific support required for the AQM project. Our EZGrid software [27, 28] provides web-based access to the grid, interfacing Globus features and providing additional support, and is described afterward.

3. Grid Portals. In general, user interaction with existing grid environments is relatively low-level. Job submission is a cumbersome process that typically requires the use of scripts and job description languages. Grid middleware tools like Globus provide command line tools to interact with grids, which are not user-friendly. The user is responsible for selecting the target platform and for describing the location of files. However, information needed in order to identify suitable target resources is usually only partially available. Grid portals assist users from the scientific community by making available a high-level interface that may provide a variety of services to support the deployment of applications on a grid. Basic features include support for identifying oneself to the grid, and displays of the information available to support resource selection, as well as forms for submitting jobs and subsequently monitoring their progress. Much more is being addressed in emerging portals, which may contain any of the following: services for resource and application discovery, construction and submission of workflow and single executable jobs, automated target resource selection (brokerage), file handling, error handling, management of projects, maintenance of data, maintenance of user profiles and access control, and more. For broad usability, a portal should be able to function in conjunction with Globus software: Globus provides interfaces known as CoG kits to the popular portal-building languages to simplify their construction. An important requirement of all portals is that they must guarantee security of user code and data.

3.1. Current portal projects. There are many efforts to provide unified web-based access to a collection of high performance computing resources. Most of them either provide a general set of grid services that are not directly applicable to complex scientific application needs, and which do not satisfy all the requirements of user-grid interactions, or they are specific to an application or organization and thus not suitable as a generic grid portal. Here, we mention just a few significant grid portal efforts.

PUNCH [29] provides users with a visualized view of computational resources that span multiple administrative domains by performing resource allocation for jobs transparently. It is thus very easy to use. But it is based on custom middleware rather than Globus, and this software must be installed on strategically distributed machines (servers) in the grid. Software is pre-installed; accredited users may run only those applications that have been made available on the grid. Also, PUNCH uses virtual accounts that require the user environment (files and data) to be transparently moved to the actual execution location. This may cause significant performance degradation in lower bandwidth environments.

Websubmit [30] is another web-based portal that gives access to applications available on a set of HPC resources while hiding the complexities of the underlying operating system. It provides form (web)-based job submission and file transfer capabilities to users. But like PUNCH, it is not based on Globus and does not benefit from components of the toolkit, such as GSI [23]. It has been implemented using CGI scripts based on Tcl that are known to

have poorer performance and portability than do Java-based web applications. Moreover, it supports only the Loadleveler batch system.

UNICORE [31] aims at providing a uniform and seamless, web-based access to grid HPC resources to carry out tasks such as job preparation, submission and monitoring. It relies on its own server modules that manage job execution on various resources. Hence, this project also does not employ Globus middleware services although interoperability initiatives are underway. UNICORE does not target provision of resource status information and requires special client and gateway software for accessing its services.

The Gateway [32] portal provides generic services such as job submission, file transfer and information services using CORBA-based [33] middleware that makes use of the services provided by Globus and a database backend. A high level Gateway API is provided that allows users to build complex tasks. But this portal also has a few drawbacks. In particular, it requires the gateway server to be running on each resource. It does not include support for resource brokerage so that a user must manually select the resources on which a job is to be run. The Gateway security infrastructure uses the CORBA standard. CORBA was not designed for high performance computing and does not provide the desired performance levels.

The XCAT [34] science portal allows grid application programmers to create scripts for complex distributed computations and to package these applications with simple interfaces for others to use. It differs from the other efforts described above, as it requires each user workstation to act as a portal with its own personal web server that is capable of running application scripts and launching remote applications for a user. For each application there is also the extra burden of packaging applications and developing an application manager (although this process is simplified by using scriptable components). Resource brokerage is not yet fully supported. GSI and MDS [35] are the only components of Globus that are used.

3.2. Toolkits for grid portals. Portals are critical for user acceptance of grids. As a result, there are already a few toolkits available to assist those developers who intend to build a grid portal. These toolkits provide the basic features required in such an effort. Here, we describe two important toolkits, GPDK [36] and GridPort [37, 38].

GPDK and GridPort/Hotpage both aim to provide generic, reusable common components to access the grid services provided by Globus. The generic services include security, simple job submission, file transfer and information services. GPDK leverages common commodity technologies such as Java beans and servlets to provide interoperability with existing technologies. It is designed using the Model-View-Controller (MVC) paradigm to provide modular and flexible portal software. GridPort is based on server side Perl modules, which have poorer performance and portability than Java-based applications, using the Perl CoG kit. The most attractive feature of GPDK is that it provides an extensible library and a template portal that can be easily extended to provide new capabilities for customized portals. However neither of these toolkits provide support for job management, resource brokerage or a broad range of information to aid the user in such tasks as selecting a target environment for a specific job. Portals such as [39, 40] have been developed based on GPDK, while [41, 42] and [43] are based on GridPort. The GRB [44] portal is similar to Hotpage portal in terms of services provided and technology used.

3.3. Portal support for AQM. A comprehensive user environment is needed by AQM and other similar projects to assist in the efficient management and deployment of various applications and computing resources. As with other early adopters of grids, there is a demand for more transparency when dealing with the execution environment, and therefore for a more comprehensive set of services at a high level. In addition to portal support for the "standard" functions provided by Globus, the AQM project has identified

the following features that are highly desired for a custom grid environment:

- The AQM project will benefit from a web-based user interface which offers easy access to computational and data resources to grid users with different roles such as researchers, administrators, and project leaders. An interface that is cognizant of the various participants, their roles and activities, can provide multiple levels of user support including but not limited to acting as a single point of access to grid and project resources. It may also permit a personalized view of resources, applications, credentials, GUI preferences and more.
- The project participants need to be able to define sequences of tasks involving meteorological forecasts, or to couple forecasting, emissions processing, and chemical transport simulation. Some of these tasks must be started at a given time. Workflow support at the user level and the ability of the underlying system to provide advance reservation of resources are thus essential features of a grid environment that will enable AQM users to conveniently and flexibly combine the different computational tasks needed to achieve such a complicated goal. A high-level interface is also necessary for the convenient definition, scheduling and submission of simple jobs as well as jobs that may entail complex dependencies between its components.
- For the AQM project, automated resource brokerage can be used to automatically and transparently select suitable resources to test the environment model configurations and find optimal setups for different geographic and seasonal conditions. By coupling with tools for data mining and data visualization, a grid environment can be enhanced to efficiently access and analyze the vast amounts of observational data and simulation results.
- For a research project such as the air quality study, in which the activities rely heavily on numerical simulation tools and computing resources, the grid environment may help the researchers improve their collaboration and help the project leaders to better manage members' research activities by recording and analyzing the usage history of the computing resources and modelling tools.

4. EZ-Grid Portal. EZ-Grid is an ongoing project to develop technologies that facilitate the utilization of grids and their supporting services, primarily by simplifying user interaction with such services and by providing automated versions of standard tasks. EZ-Grid software can be used to build a generic interface for working on a grid, or it can be used to create a custom grid environment. The project also aims to provide a more complete set of user-level grid services in accordance with the demand for simplification of grid activities.

4.1. Basic functionalities. The EZ-Grid portal provides the following basic services required for user interaction with the grid, in a graphical environment:

- *Job specification and submission*: Means for the user to specify and submit a job, including the location of input and output files and compute needs; arguments may be passed to SGE in our local setup. Automated translation of these into Globus RSL is followed by job submission to GRAM.
- *Job management*: Storage and retrieval of relevant user and application profile information, history of job executions and related information. Application profiles consist of metadata that characterize frequently run jobs.
- *File handling*: Transparent authentication with and browsing of remote file systems that are part of the grid resources is supported. Data can be securely transferred between grid resources using the GSI-enabled data transport services and GridFTP.
- *Data archive maintenance*: this feature allows users to organize, locate and use the huge amount of data typically produced by scientific applications.
- *Single sign-on*: Globus proxy (temporary identity credential) creation using Grid Security Infrastructure and X509 certificates. This allows the user to seamlessly

establish his or her identity across all campus grid resources by providing his portal identity and passphrase. Upon sign-on, the user's Globus proxy is generated on the portal server; and used for subsequent authentication with the grid resources using GSI. All communications are carried out over secure `http` channels.

Our campus grid's credential server provides a repository for user credentials, such as the X509 certificate [45] and key pairs and proxies. In our current campus grid setup, we have adopted a stand-alone credential server model to allow users to access grid services with unlimited mobility. Alternatively, a MyProxy [46] server can be used to act as an online repository for user proxies. However, this adds an upper limit to the mobility of the users, due to the limited lifetime of the Globus proxies delegated. Appropriate mapping of the portal accounts to proxies allows users to perform single sign-on and to access grid resources. The portal supports the export of encrypted keys from the user to the credential server using secure `http` sessions. In some scenarios, the credential server can also be used to generate the user credentials, thus ensuring enhanced security by not exposing the private keys.

4.2. Advanced functionalities. Here we describe several features that are not yet provided by other existing grid portals but which are necessary for scientific applications. The first three are offered by the EZ-Grid portal and the remaining two are currently under development:

- *Comprehensive resource information*: Status information on grid resources, both static (hardware and software information) and dynamic attributes (CPU loads, job queue and status information). Static information is obtained primarily from Globus Information services (MDS); dynamic scheduler information and queue details are retrieved from SGE using our information subsystems. Additional information provided graphically includes application profiles and job execution histories, as mentioned above.
- *Resource brokerage*: automatic and transparent selection of a set of available target resources that match the user job requirements. The resource brokerage uses the resource information, the usage control services (see below) and the scheduler specific interfaces to make resource choices based on the job specifications. The goal of this brokering process could be budget-based or deadline-based or both. Accordingly, the broker is expected to choose resources with the lowest cost or with the fastest execution time. The brokering process is split into a series of stages in which the potential set of resources is progressively narrowed down to ultimately identify the best resource for the given job specification. It starts by filtering based on static resource information, including the application profile, which needs no interaction with the remote resources. Then, the policy-based evaluation determines which resources are available to the user and the cost; this too incurs little or no interaction with remote resources. After these initial stages are carried out, network parameters such as available bandwidth and latency are assessed; these are critical if large datasets must potentially be transferred. For those resources that are thus considered most likely candidates, any additional history information about previous job submissions is fetched from the remote resource and analyzed. Finally, the scheduler interfaces at the remote resource are queried to obtain dynamic information about the work queues and the current loads in order to estimate job run times.
- *Precise usage control*: Policy-based authorization and accounting services to assist in examining and evaluating usage policies of the resource providers. Policies on system usage set up and enforced by the provider may focus on diverse issues such as security, priority, class of acceptable jobs and users, quality of service and avoidance of misuse. They may need to be flexible; for example, it may be appropriate

to modify the priorities for a particular job according to changes in the dynamic environment. Tools are provided for the administrators of grid resources to specify and enforce policies at the grid environment level. The user may view this information while making job submissions [47]. The policies are part of the information that is evaluated as part of the brokerage process.

- *Customizable user profile*: a customizable view of resources, applications and personalized contents. Personalized user information includes not only basic information such as the name, email address etc. but also the desired display of information about resources, credentials, portal GUI preferences and settings associated with the user. We are modifying our EZ-Grid portal so that it will be able to provide user-specific GUI customization, including desired color palette and background colors, help information for personal interests, and so on.
- WorkFlow management: support for the specification of multi-executable jobs, where the executables interact in a potentially non-trivial manner. The AQM work requires workflow support so that the different parts of jobs with multiple codes can be started at appropriate times. Current work focusses on building a GUI that helps users to create, store, run and monitor complex tasks and applications. We intend to enable the user to graphically construct workflow jobs via easy drag-and-drop operations. Workflow specifications can then be stored and scheduled, as needed, to automatically run jobs periodically or at a specified time. We plan to leverage recent and developing technologies such as GridAnt [48] for realization of the workflow management.

4.3. EZ-Grid implementation. The two most common technologies for building web-based applications are CGI/PERL and JSP/Servlets/Java Beans. Traditionally, web applications were built using CGI/PERL technology. CGI is a standard for interfacing external programs with information servers such as a web server. The server submits client requests encoded in URLs to the appropriate registered CGI program, which executes them and returns results back to the server. CGI programs are executable programs that run on the web server; although they are usually written in PERL, they can be written in any available scripting language or programming language that can run on a web server, including PERL, TCL, and Unix shells. A grid portal can be developed by writing CGI programs that make use of Globus PERL COG kit scripts. An alternative to this approach is to use Java technologies such as servlets, Java beans and JSP. Java Servlet technology provides web developers with a simple, consistent mechanism for extending the functionality of a web server based on the power of the Java language. It offers a component-based, platform-independent method for building web-based applications. The EZ-grid implementation is based on Java Servlets because they have the following advantages over traditional CGI applications:

- Efficiency: With CGI, a new program is created for every `http` request, but with Java servlets a new thread is spawned instead of a new program for every request. This approach reduces server memory requirements and saves time by instantiating fewer objects.
- Enhanced functionality: Java servlets may exploit the power of the Java programming language and talk directly to the web server, whereas regular CGI programs cannot, at least not without using a server-specific API. Communicating with the web server makes it easier, for instance, to translate relative URLs into concrete path names. Multiple servlets can also share data, making it easy to implement database connection pooling and similar resource-sharing optimizations. Servlets can also maintain information from request to request, simplifying strategies for session tracking and caching of previous computations.

- Portability: Java servlets are more portable than CGI programs because they are written in Java's standard API. Servlets are supported directly or by a plug-in on virtually every major web server.
- Security: CGI programs are mostly executed as shell scripts and different shells may treat symbols like '\' differently. Any errors caused by inappropriate use of such symbols may lead to unexpected results and server crashes. Also, Java has more stringent checks for memory binding and types than is performed for CGI.
- Convenience: Java servlets provide more convenient methods for reading and setting `http` headers, handling cookies, tracking sessions, and many other such high-level utilities.

The Apache Struts Web Framework [49] encapsulates the MVC paradigm to provide structures and common web functionalities that greatly speed up web application development with JSP/Java Beans and improve the performance of the final product. JSP [50] provides multiple ways to use templates in web-site building. In EZ-Grid, the web-page template is maintained in a single place and is included dynamically upon viewing, allowing the web application front end to be easily integrated with any web portal interface design. Currently, the EZ-Grid web application contains more than 40 JSP files. However, these files use only two design templates. If the portal is to be customized for use by an institute, company or project, the web pages must be redesigned to incorporate the desired user background and meet specified needs. This might include, for example, a logo, help information or adding a news section to the web pages with articles that are closely related to their research topics. Unless care is applied, it might be necessary to revise the whole 40 JSP pages to suit the needs of this project. But, as the EZ-Grid portal deploys a dynamic template built using Struts template tag library, which makes the layouts of web pages much more flexible and maintainable, the new portal can be developed just by changing just a few layout files. Figure 4.1 contains some sample JSP code segments that show how easy it is to customize the portal's web page appearance to suit particular user needs using this dynamic template technique. The webmaster who deploys or maintains the portal can easily customize it by designing or updating one or two template files; in the example, the file modified is *template.jsp*. The webmaster needs to make sure that the following three header lines required by EZ-Grid appear as part of the JSP headers for every new template he or she creates

```
<%@ taglib uri="/template" prefix="template" %>
<%@ taglib uri="/WEB-INF/app.tld" prefix="app" %>
<app:checkLogon/>
```

and then use the `template:get` tags provided by the Struts framework to control which part of the EZ-Grid portal interface should go into the design template. No other JSP pages ever need to be changed. The EZ-Grid related tags are underlined in the figure. The first of these is `<template: get name="EZ-GridJavaScripts"/>`, which denotes the EZ-Grid related Javascripts. The remaining three denote the header, content (forms etc. implementing the actual EZ-Grid functionalities) and EZ-Grid related footer respectively. This facility gives the webmaster the ability to change the appearance of the web-pages while retaining the EZ-Grid portal functionalities. It is important to notice that, even though all the template files are JSP files, the webmaster does not need to know JSP and can use any popular web development tool such as Dreamweaver to design the template page.

Figure 4.2 shows how the EZ-Grid architecture is layered and also indicates its interaction with other middleware tools and resource management systems. The presentation layer consists of HTML/JSP web pages containing a customizable GUI to provide users with convenient and useful graphical interfaces. The web application service layer uses the Struts framework to organize/model the generic web application services in the standard MVC paradigm. The so-called business logic consists of Java classes implementing the ac-

File : template.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
import="java.sql.*"%>
<%@ taglib uri="/template" prefix="template" %>
<%@ taglib uri="/WEB-INF/app.tld" prefix="app" %>
<app:checkLogon/>
<html>
<head>
<title>AQMG   EZ-Grid Portal</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<template:get name="EZ-GridJavaScripts"/>
</head>

<body>
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr>
    ... more code   ...
    <td width="72%">
      <template:get name="EZ-GridHeader"/>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <template:get name="EZ-GridContent"/>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <template:get name="EZ-GridFooter"/>
    </td>
  </tr>
</table>
</body>
</html>
```

File : editJobProfile.jsp

```
<%@ taglib uri="/template" prefix="template" %>
<template:insert template="template.jsp">
  <template:put name="EZ-GridJavaScripts" content="jobprofilejs.htm"/>
  <template:put name="EZ-GridHeader" content="appheader.htm"/>
  <template:put name="EZ-GridContent" content="mainMenuPart.jsp"/>
  <template:put name="EZ-GridFooter" content="appfooter.htm"/>
</template:insert>
```

FIG. 4.1. *Dynamic templates in EZ-Grid*

tual services required by the EZ-grid portals such as single sign-on, job submission, file transfer, etc. The business logic makes use of the services provided by the Grid Service Interface, which is actually a layer that consists of a standardized encapsulation of the grid services provided by the Globus toolkit. Finally, the Database access layer provides methods for accessing databases as needed. Note that although this is not shown in the diagram, EZ-Grid interacts both directly and indirectly (via Globus) with the resource manager.

4.4. OGSI. With the growing interest in grids and their commercial development and deployment, the lack of accepted standards for interoperability between diverse grid computing technologies used in individual organizations [51] became a major impediment to progress. The integration of dynamic business applications in the form of web services was sought. In a grid environment, means must be provided to support the dynamic discovery of services and to exploit other related tools. The Open Grid Services Architecture (OGSA) provides a generic grid system architecture that leverages web services technologies and enhances existing grid technologies and tools to realize "grid services". Such grid services are based, though not exclusively, upon SOAP, WSDL and other web services concepts to enable wide-scale resource sharing with no tight coupling to any specific protocols, thereby avoiding associated deficiencies. OGSI is a concrete technical specification being drafted by the OGSI

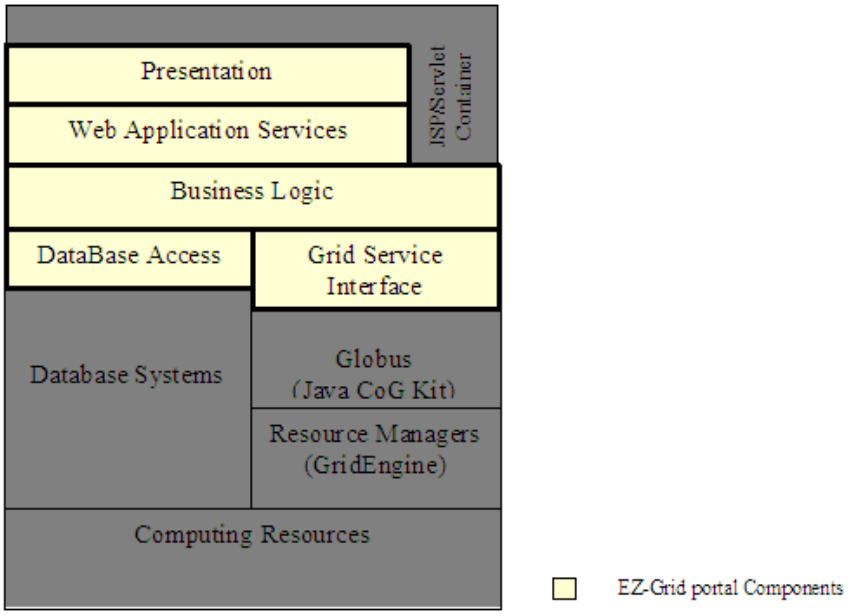

FIG. 4.2. *EZ-Grid Portal Architecture*

working group of the Global Grid Forum (GGF) that provides the basic conventions for implementing grid services. It defines mechanisms for creating, managing and exchanging information among grid services. A grid service should be an addressable instance that has facilities to achieve its creation and its information introspection, manage its lifetime, and provide for its invocation. Grid service handles (GSH) (global URI names) and Grid service references (GSR) (WSDL document with the specifics of the bindings to facilitate service invocation) facilitate naming, identifying and invoking specific grid service instances. Grid services may be persistent or transient. Mechanisms have been defined for soft-state lifetime management to enable authorized clients to extend, shorten or immediately destroy grid service instances. Typically, client applications/programs will be isolated from the web service invocation by an interface that acts on behalf of the client to perform argument marshalling, message routing and invoking appropriate bindings on the server side.

Version 3 of the Globus toolkit (GT3) provides a reference implementation in Java for the current grid service specifications and a container to create and invoke basic Globus toolkit services such as GRAM and GSI or other OGSI compliant grid services. The core of GT3 provides the basic OGSI functionality to enable reuse by grid service programmers. The client and the server side implementations are JAX-RPC [52] compliant and the stubs for the service interfaces are generated using automated tools. This makes it relatively straightforward for a programmer who knows the conventions to implement and offer additional OGSI-compliant services.

4.4.1. OGSI compliance in EZ-Grid. The OGSI specification has defined a few portTypes that can be extended to create OGSI-compliant Grid services. They capture the most common behavior expected from distributed computing components. The service interfaces are translated into WSDL specifications (portTypes) and are used to generate stubs for the clients for introspection and invocation.

Put simply, the designer of OGSI-compliant components should combine and extend a

combination of GridService portTypes (a base interface definition that captures the behavior expected of all grid services), optional portTypes (such as Factory, ServiceGroup and so on) and other application-specific portTypes. OGSI-compliant services should inherit from the ServiceSkeleton class (that is part of the GT3 core) and implement the WSDL portTypes exposed in the interface specification of the service as well. The major portal services of EZGrid such as information services, job submission and monitoring services have been reengineered to be compliant with the OGSI specification. Interfaces have been created to support lifetime management, state introspection and more, and they inherit from the ServiceSkeleton class as required. The interface with Sun Grid Engine complies with the Master Managed Job Manager Factory Service instances (MMJFS) and job manager instances of GT3.

OGSA and the related specifications are still undergoing refinement and are therefore subject to change. We are closely monitoring GGF developments to ensure that our portal services will be compliant with the most recent specifications for grid services.

5. Related Work. There are several other air-quality or meteorological studies which have begun to take advantage of grid technologies. The Modelling Environment for Atmospheric Discovery project [53, 54] supported by NCSA focuses on retrospective analysis of hurricanes and severe storms using the TeraGrid [55], integrating computation, grid workflow management, data management, model coupling, data analysis/mining, and visualization. CrossGrid [56] meteorological applications [57] analyze archived operational data from a mesoscale model. Meteorological reanalysis of databases, which include homogeneous meteorological information from a single numerical weather prediction model integrated over decades, will also be performed. The system will include an atmospheric pollution chemistry module.

6. Conclusions and Future Work. Grid environments enable new infrastructures such as "virtual organizations" that seamlessly aggregate distributed and dynamic collections of resources. Campus grids provide good opportunities for researchers to solve computationally demanding problems in science and engineering. However, simplified access to grid services is essential for computational scientists to fully utilize the advantages. Grid portals enable users to gain web-based access to grid resources and services. They may be used to perform single sign-on, view resource status information, submit jobs and manage data.

The EZ-Grid portal project addresses these issues and provides a single web-based interface for performing standard grid tasks. The adaptation of the portal to meet specific project or individual user interface needs is relatively straightforward. It has enabled researchers in our AQM project to automate the process of specifying and starting complex jobs that utilize a variety of hardware resources on our campus. Ongoing work extends our current functionality to support additional needs of this project and of other computational projects at our university. Future work includes development of strategies for increased fault-tolerance and maximum performance gains to meet the needs of time-critical applications. With an increasing user demand for insulation from the complexities of a heterogeneous and distributed grid environment, portals such as EZ-Grid and their services will become increasingly sophisticated.

HiPCAT is a consortium of leading research institutions in Texas that share advanced technologies and resources to simulate new methods of computing and collaboration. The University of Houston participates in this initiative. The future HiPCAT portal will provide authenticated users with information on the grid resources, including static and dynamic information on their status and the job queues. We intend to collaborate with our HiPCAT colleagues so that EZ-Grid functionality may be part of this environment and AQM applications can be deployed in this larger computational grid.

Acknowledgements. This work is being performed as part of the Sun Center of Excellence in Geosciences. We wish to thank our colleagues within this Center and the staff of the HPC Center for their support for this initiative. We would also like to thank Sun MicroSystems^{TM} for their support and guidance.

REFERENCES

[1] I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, in International Journal of Supercomputer Applications, 15(3), 2001.

[2] I. Foster and C. Kesselman, *Globus: A metacomputing infrastructure toolkit*, in International Journal of Supercomputer Applications 11(2): 115-128.

[3] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, Paul F. Reynolds Jr, *Legion: The Next Logical Step Toward a Nationwide Virtual Computer*, Technical Report No. CS-94-21. Department of Computer Science, University of Virginia. June 1999

[4] Institute for Multidimensional Air Quality Studies (IMAQS): http://www.aqm.uh.edu

[5] I. Foster, C. Kesselman, J. Nick and S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Globus Project, 2002, www.globus.org/research/papers/ogsa.pdf.

[6] *EPA: Guideline for Developing an Ozone Forecasting Program* `http://www.epa.gov/ttn/oarpg/t1/memoranda/foreguid.pdf`

[7] P. C. Manins, M. E. Cope, G. D. Hess, P. F. Nelson, K. Puri, N. Wong and M. Young, *The Australian Air Quality Forecasting System: prognostic air quality forecasting in Australia*, Clean Air, 36 (2): 43-48.

[8] *Mesoscale Model (MM5)*, Pennsylvania State University, National Center for Atmospheric Research, `http://www.mmm.ucar.edu/mm5`

[9] J. Dudhia and J. Bresch, *A global version of the PSU-NCAR mesoscale model*, Mon. Wea. Rev., 130: 2989-3007, 2002.

[10] G. Grell, J. Dudhia, and D. Stauffer, *A Description of the Fifth-Generation Pennsylvania State/NCAR Mesoscale Model (MM5) NCAR/TN-398+STR*, NCAR Tech Notes

[11] J. W. Nielsen-Gammon, *Initial Modeling of the August 2000 Houston- Galveston Ozone Episode*, A Report to the Technical Analysis Division, Texas Natural Resource Conservation Commission, December 19, 2001

[12] J. Michalakes, J. Dudhia, D. Gill, J. Klemp and W. Skamarock, *Design of a next- generation regional weather research and forecast model: Towards Teracomputing*, World Scientific, River Edge, New Jersey, 117-124, 1998.

[13] J. Michalakes, *Weather Research and Forecast Model 1.2: Software Design and Implementation*, internal draft documentation, October 2002.

[14] *Sparse Matrix Operator Kernel Emissions (SMOKE) Modeling System*, `http://www.emc.mcnc.org/products/smoke/`

[15] Z. Adelman and M. Houyoux, *Processing the National Emissions Inventory 96 (NEI96) version 3.11 with SMOKE*, the Emission Inventory Conference: One Atmosphere, One Inventory, Many Challenges, 1-3 May, Denver, CO, U.S. Environmental Protection Agency, 2001.

[16] Vukovich, J. McHenry, C. Coats and A. Trayanov, *Supporting Real-Time Air Quality Forecasting using the SMOKE modeling system*, Denver, CO, EPA Emissions Inventory Conference, April 30-May 2, 2001.

[17] *Comprehensive Air quality Model with extensions (CAMx)*, `http://www.camx.com/`

[18] D. Byun, J. Pleim, R. Tang, and A. Bourgeois, *Meteorology-Chemistry Interface Processor (MCIP) for Models-3 Community Multiscale Air Quality (CMAQ) Modeling System*, Washington, DC, U.S. Environmental Protection Agency, Office of Research and Development.

[19] K. Czajkowski,I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S. Tuecke, *A Resource Management Architecture for Metacomputing Systems*, in Proceedings of the IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, USA, 30 March 1998.

[20] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel and S. Tuecke, *Data Management and Transfer in High-Performance Computational Grid Environments*, In Proceedings of the Parallel Computing 2001, Naples, Italy, 4-7 September 2001.

[21] J. Bester, , I. Foster, C. Kesselman, J. Tedesco and S. Tuecke, *GASS: A Data Movement and Access Service for Wide Area Computing Systems*, Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems, Atlanta, USA 5 May 1999.

[22] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, *Grid Information Services for Distributed Resource Sharing*, In Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, San Francisco, USA, 7-9 August

2001.
[23] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, *A Security Architecture for Computational Grids*, Proceedings of 5th ACM Conference on Computers and Security, San Francisco, USA, 3-5 November 1998.
[24] *Simple Object Access Protocol (SOAP), Version 1.2*, http://www.w3.org/TR/soap12-part0/
[25] *Web Services Description Language (WSDL), Version 1.2*, http://www.w3.org/TR/wsdl12/
[26] *Sun Grid Engine, Sun Microsystems 2002*, http://www.sun.com/software/gridware
[27] B. M. Chapman, J. He, Y. Li and B. Sundaram, *A Computational Environment for Air Quality Modeling in Texas*, In: W. Zwieflhofer (Ed.), Use of High Performance Computing in Meteorology, World Scientific Publishing Co., 2003
[28] B. M. Chapman, H. Donepudi, J. He, Y. Li, P. Raghunath, B. Sundaram and Y. Yan, *An OGSI-Compliant Portal for Campus Grids*, 10th ISPE Int. Conf. on Concurrent Engineering, Portugal, July 2003 (to appear)
[29] N. H. Kapadia, J. A. B. Fortes, M. S. Lundstrom, and D. Royo, *PUNCH: A Computing Portal for the Virtual University*, International Journal of Engineering Education (IJEE). In special issue on Virtual Universities and Engineering Education. Vol. 17, No. 2.
[30] J. E. Koontz, R. P. McCormack, and J. E. Devaney, *WebSubmit: a paradigm for platform independent computing*, Proceedings of the Workshop on Seamless Computing, Reading, England, September, 1997.
[31] M. Romberg, *The UNICORE architecture: Seamless access to distributed resources*, Proceedings of the 8th International Symposium on High Performance Distributed Computing HPDC-8, August 1999.
[32] T. Haupt, E. Akarsu, G. Fox, A. Kalinichenko, K.-S. Kim, P. Sheethalnath, and C.H. Youn, *The Gateway System: Uniform Web Based Access to Remote Resources*, In ACM 1999 Java Grande Conference, pages 1–7, San Francisco, CA, June 1999.
[33] *CORBA: Common Object Request Broker Architecture*, http://www.corba.org.
[34] S. Krishnan, R. Bramley, D. Gannon, M. Govindaraju, R. Indurkar, A. Slominski, B. Temko, R. Alkire, T. Drews, E. Webb, and J. Alameda, *The XCAT Science Portal*, Proceedings of Supercomputing 2001
[35] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, *Grid Information Services for Distributed Resource Sharing*, Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing, August 2001
[36] *Grid Portal Development Kit (GPDK)*, http://www-itg.lbl.gov/grid/projects/GPDK/
[37] *SDSC GridPort Toolkit*, http://gridport.npaci.edu/
[38] M. Thomas, S. Mock, and J. Boisseau, *Development of Web Toolkits for Computational Science Portals: The NPACI HotPage*, The 9th IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, August 2000.
[39] NASA Information Power Grid, https://portal.ipg.nasa.gov
[40] M. Russell, G. Allen, G. Daues, I. Foster, T. Goodale, E. Seidel, J. Novotny, J. Shalf, W. Suen, and G. Von Laszewski, *The Astrophysics Simulation Simulation Collaboratory: A Science Portal Enabling Community Software Development*, Proceedings of Tenth IEEE International Symposium on High Performance Distributed Computing, August 2001(submitted)
[41] *LAPK Portal: The laboratory for Pharmacokinetics Modeling*, https://gridport.npaci.edu/LAPK
[42] *GAMESS Portal The laboratory for Pharmacokinetics Modeling*, https://gridport.npaci.edu/GAMESS
[43] *The DSG grid portal web page*, https://l59.csm.port.ac.uk/dsgPortal/
[44] G.Aloisio, M.Cafaro, P.Falabella, C.Kesselman and R.Williams, *"Grid Computing on the Web using the Globus Toolkit"*, Proc. HPCN Europe 2000, Amsterdam, Netherlands, Lecture Notes in Computer Science, Springer-Verlag, N. 1823, 32-40, 2000
[45] S. Tuecke, D. Engert, I. Foster, M. Thompson, L. Pearlman and C. Kesselman, *Internet X.509 Public Key Infrastructure Proxy Certificate Profile*, IETF, Draft draft-ietfpkix-proxy-01.txt, 2001.
[46] J. Novotny, , S. Tuecke and V. Welch, *Initial Experiences with an Online Certificate Repository for the Grid: MyProxy*, The Tenth IEEE International Symposium on High Performance Distributed Computing, August, 2001
[47] B. Sundaram and B. M. Chapman, *Policy Engine: A Framework for Authorization, Accounting Policy Specification and Evaluation in Grids*, 2nd International Conference on Grid Computing, Nov 2001.
[48] S. Krishnan, P. Wagstrom and G. von Laszewski, *GSFL: A Workflow Framework for Grid Services*, In Preprint ANL/MCS-P980-0802, Argonne National Laboratory
[49] *The Apache Struts Framework web page*, http://jakarta.apache.org/struts.
[50] *The Java Server Pages web page*, http://java.sun.com/products/jsp
[51] I. Foster, C. Kesselman, J. Nick and S. Tuecke, *Grid Services for Distributed System Integration*, Computer, 35(6), 2002. http://www.globus.org/research/papers/ieee-cs-2.pdf
[52] *Java API for XML Processing 2001*, http://java.sun.com/xml/jaxp/index.html

[53] R. Wilhelmson, K. Droegemeier, S. Graves, M. Ramamurthy, D. Haidvogel, B. Jewett, J. Alameda and D. Gannon, *Modeling Environment for Atmospheric Discovery. MEAD*, Preprint - Annual American Meteorological Society Meeting - February, 2003
[54] *MEAD*, http://www.ncsa.uiuc.edu/expeditions/MEAD/
[55] *TeraGrid Project*, http://www.teragrid.org/
[56] *CrossGrid project*, http://www.crossgrid.org
[57] *CrossGrid Meteorological Applications*, http://www.eucrossgrid.org/Presentations/IST2002_meteo.zip