Predicting Air Quality in a Production-Quality Grid Environment

Barbara M. Chapman, Priya Raghunath, Babu Sundaram, and Yonghong Yan

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-05-04

February 22, 2005

Keywords: Air Quality Forecasting (AQF), Campus Grid, Grid Resource Management, Metascheduler, Authentication

Abstract

Air Quality Forecasting (AQF) is a new computational discipline that attempts to predict atmospheric pollution, especially high levels of ozone. The application is complex, incorporating weather models, emissions processing and chemical transport models at multiple levels of refinement; it poses substantial computational and storage requirements. Deployment in a grid is one way in which timely and reliable production of forecast results may be ensured. We have extensively studied an AQF application based upon a community Air Quality model in order to determine its development and execution requirements as well as the ability of current grid technology to satisfy all phases of preprocessing, model execution, and storage and retrieval of observational and generated simulation data. A production-quality campus grid is being built at the University of Houston using up-to-date grid software to support this application. In this chapter, we discuss AQF and its computational needs, current grid-building software, and our experiences using it to build the campus grid. We describe the shortcomings of existing grid middleware that were identified during the course of this work and present our efforts to augment available software and to overcome some of these problems, with a focus on the user environment, resource management and authentication issues.

Predicting Air Quality in a Production-Quality Grid Environment

Barbara M. Chapman, Priya Raghunath, Babu Sundaram, and Yonghong Yan

Abstract

Air Quality Forecasting (AQF) is a new computational discipline that attempts to predict atmospheric pollution, especially high levels of ozone. The application is complex, incorporating weather models, emissions processing and chemical transport models at multiple levels of refinement; it poses substantial computational and storage requirements. Deployment in a grid is one way in which timely and reliable production of forecast results may be ensured. We have extensively studied an AQF application based upon a community Air Quality model in order to determine its development and execution requirements as well as the ability of current grid technology to satisfy all phases of preprocessing, model execution, and storage and retrieval of observational and generated simulation data. A production-quality campus grid is being built at the University of Houston using up-to-date grid software to support this application. In this chapter, we discuss AQF and its computational needs, current grid-building software, and our experiences using it to build the campus grid. We describe the shortcomings of existing grid middleware that were identified during the course of this work and present our efforts to augment available software and to overcome some of these problems, with a focus on the user environment, resource management and authentication issues.

Index Terms

Air Quality Forecasting (AQF), Campus Grid, Grid Resource Management, Metascheduler, Authentication

I. Introduction

Air Quality Forecasting (AQF) [19] is a recent discipline that addresses important air pollution problems and attempts to provide a basis for dealing with them. With the increasing maturity of Air Quality Models, air quality forecasting services are beginning to be established. We participate in an effort to build such a service, with the goal of providing timely, reliable forecasts of air quality for the Houston-Galveston region and for several other regions in the South Central USA that have encountered problems with air quality in the recent past. On-going work at the University of Houston (UH) aims to create, test and deploy an AQF application [49] as well as to establish a suitable development and deployment environment. The application cycle places substantial demands on this execution environment. It makes intensive use of sophisticated numerical tools, requires high compute power for the numerical simulation of meteorological and chemical processes, and entails the transfer, storage and analysis of a huge amount of observational and simulation data [9].

The recent emergence of computational grid technology [22], and middleware tools to enable the creation of such grids, provides a potential strategy for meeting the computational and storage needs of AQF codes. Grid technology permits the creation of virtual organizations [25] that provide reliable access to (potentially) widely distributed computing resources and enables seamless resource sharing among groups of institutions and between individuals. Grid software enables applications to exploit computational and information resources that may be owned and managed by distinct organizations with diverse usage policies. Current grid-building initiatives have a variety of goals, from targeting the needs of a specific application or class of applications to maximizing system throughput across an organization or group of institutions. Users with large-scale problems, such as AQF applications, may exploit multiple distributed high performance computing resources in a grid environment to run individual, complex jobs.

In a close collaboration between AQF researchers and Computer Scientists at UH, a customized computational grid environment is being created to fulfill the needs of our air quality forecast project. The work performed builds upon results of the EZ-Grid Project [12], which researched and developed grid infrastructure for higher-level interaction with grids by end users and system administrators. It is supported by Sun Microsystems as part of a Center of Excellence in Geosciences based at UH. In this chapter, we discuss the state of the art in Air Quality

Forecasting, report upon our efforts to construct a production-quality campus grid [13] at UH that supports the needs of the AQF application and on our work to develop the associated portal environment. We explain the role played by existing grid middleware (especially the Globus toolkit [21], which is the de-facto standard middleware for grids), the need for additional infrastructure to deal with application-specific workflow requirements, and our effort to simplify the use of the grid, with a focus on authentication issues.

The remainder of the chapter is organized as follows. Section 2 describes current efforts to provide Air Quality Forecasting services and gives details on the on-going work at UH, which focuses on the provision of such services for the Houston-Galveston region of Texas. In Section 3, we briefly review campus grid technology including Globus, describe the UH Campus Grid and the EZ-Grid portal, a light-weight grid user/administrator interface deployed on it. Section 4 analyzes the specific needs of the AQF project with respect to grid deployment. Our evaluation of existing software with respect to these needs, and our efforts and experiences in realizing a production-quality campus grid for air quality prediction in the Houston area are discussed in detail in Section 5. Finally, we present our conclusions and indicate future work in Section 6.

II. Computational Air Quality Forecasting

Physical and chemical processes in the atmosphere are complex, coupled and occur on a wide range of scales. Air Quality Modeling studies such interactions, including the formation and dispersion of a variety of species, and air quality forecasting attempts to predict the occurrence of ozone and other relevant sources of air quality problems. Multi-scale models are needed to model and perform forecasting in urban areas [92]. An Air Quality Model (AQM) [55] must be coupled with a high-resolution weather forecast, as well as pollutant emission processing, to produce AQF results. It is therefore a major computational challenge to achieve the required high resolution results while producing model output in a timely fashion. Recently, the U.S. Weather Research Program, National Oceanic and atmospheric Administration (NOAA) [80] and EPA [60] announced that air quality forecasting is one of the key research areas that require further operational developments [19]. Several AQF activities have been initiated by these and other local entities.

As the leading organizations for the establishment of operational air quality forecasting services, the NCEP [77] /NOAA and NERL [79] /EPA (National Exposure Research Laboratory) are collaboratively integrating the EPA's Community Multiscale Air Quality (CMAQ) model [8][10] and NCEP's ETA weather model to provide regional air quality forecasting for the Northeastern U.S.A. Unfortunately, there are no current plans to provide similar regional air quality forecasting services for other parts of the US. Yet there is a need to establish air quality forecasting operations for urban areas that are subject to air quality problems. In Texas, the Houston-Galveston, Beaumont-Port Arthur, and Dallas-Fort Worth areas are all classified as Federal ozone non-attainment areas, i.e. have violated national air quality standards. Houston and Dallas are among the largest cities in the nation, and prediction of ozone violations in these local areas is urgently needed.

A. AQF Efforts at the University of Houston

The Institute for Multidimensional Air Quality Studies (IMAQS) at the University of Houston (UH) is a multi-disciplinary center involving environmental scientists, chemists, mathematicians and computer scientists including ourselves, that analyzes the air quality in the Houston area in order to provide solutions for local air quality problems. It leads the AQF project, which aims to establish air quality forecasting operations for the three ozone non-attainment regions in Texas.

To achieve this, IMAQS is building an integrated computational model for regional and local air quality forecasts that is composed of three subsystems: the PSU/NCAR MM5 mesoscale weather forecast model [27], the Sparse Matrix Operator Kernel Emission System code (SMOKE) [1], and EPA's CMAQ chemical transport model. These components may execute within multiple heterogeneous computing environments, with the start time of each module dependent on the completion of one or more prior modules. Initial meteorological conditions are provided by the daily ETA forecast analysis, the availability of which thus limits the starting time for the forecast run. In order to achieve the desired results, a very high resolution limited area weather forecast must be computed over the regions of interest. This is enabled by running a sequence of MM5 weather models with increasing resolution and decreasing geographical boundaries. The lower resolution model results provide initial boundary conditions for higher resolution regional and local model runs, as well as their periodic refreshment. Therefore, meteorological,

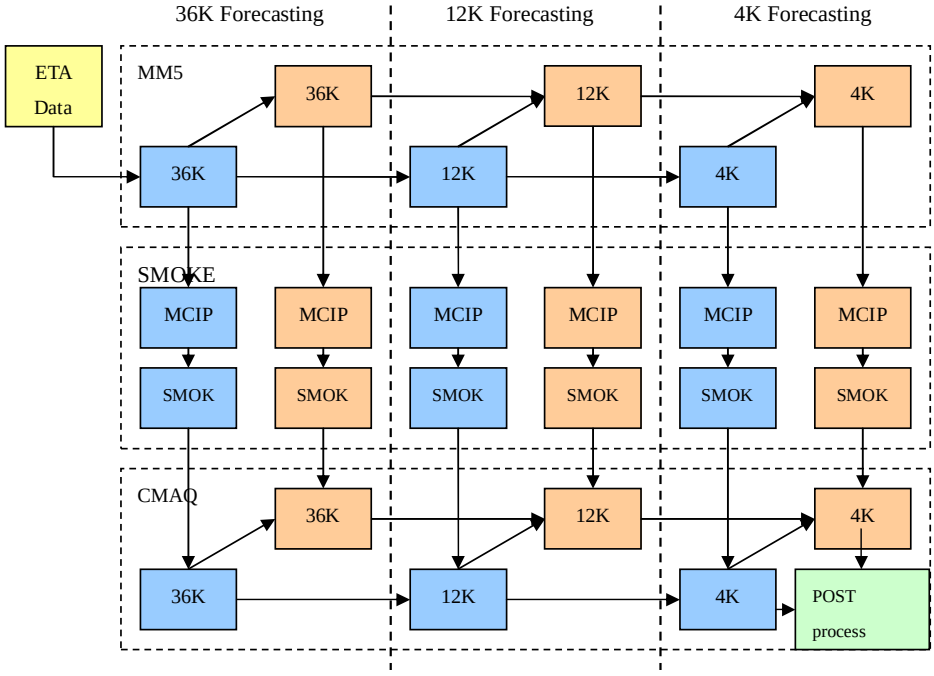

Fig. 1. Workflow Detail of AQF Application

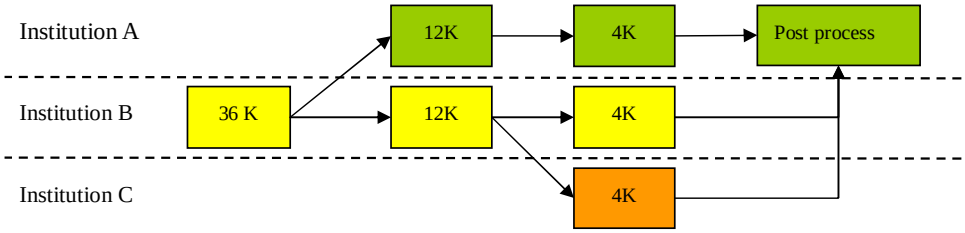

Fig. 2. AQF Execution Across Institutions

emission inventory, air quality monitoring and model simulation data need to be transferred repeatedly among the several modeling subsystems that are part of the overall application. The executables may potentially execute on different platforms (parallel or sequential), possibly at different geographic locations, so long as output is reliably transferred between them. Finally, in addition to a timely broadcast of daily model output, the results of individual runs must be stored and made available to researchers and state and local officials to study the patterns of air quality and its relationship to weather conditions and emission scenarios. Fig 1 illustrates the workflow of the nested forecasting operation for the output of an air quality forecast over a single region of interest. In the diagram, each rectangle represents an executable and each arrow indicates the flow of data. Output is produced hourly, so that after one executable has begun to produce data, its successors in this workflow may begin their computation. The 36km grid provides coarse forecast data over a large area of the earth's surface (in our case, the continental USA), the 12km grid provides data across a portion of this area (here, the south central USA), and the 4km grid forecasts air quality across a smaller geographic region of interest contained within the area covered by the 12km grid, and very detailed local topographical information. A full computation to forecast air quality in an urban area requires an additional level of refinement based upon a 1km grid.

The goal of the AQF project is to forecast air quality in the three urban regions of interest in Texas. Note that it is possible to reduce the amount of computation required when performing air quality forecasts for multiple geographic regions that are "nearby". Fig 2 shows how three institutions might each perform local forecasts, whereby only one of them completes an entire AQF run. A second institution is able to exploit the output of the 36km run and a third institution computing a forecast in an area that is covered by the same 12km grid need only begin computation at

the 4km grid level. Longer-term goals are to extend the work described here in order to enable local entities to perform their own forecast, but to reduce the amount of overall computation by exploiting results in the manner indicated. A multi-institution computational grid is expected to provide the computational infrastructure for this application cycle.

III. Campus Grids and The EZ-Grid Portal

The term ***campus grid*** is used to denote the sharing of computational resources across an organization such as a university, where the resources are often distributed among different departments and buildings and administered by different groups. They are typically used to solve large problems in science and engineering. In contrast to multi-organizational wide area grids, in such a setup there is the potential for making organizational decisions on deployable infrastructure, procedures, account naming and granting mechanisms, security and operational policies, which is likely to facilitate resource sharing. In addition to the coordination of system administration efforts, it may be possible to establish an organization-wide approach to providing and monitoring essential components, such as the network infrastructure, across the entire grid. Some campus grids are essentially collections of computer hardware that share a single file system, e.g. via NFS. Increasingly, however, such campus grids do not rely on shared files but employ some or all of the functionalities of the Globus toolkit [23] to enable the exploitation of resources across several (sub)domains of the Internet. Several research institutions including our own are early adopters of this approach [65][67][75][93].

The Globus project was instrumental in developing protocols and services for constructing grids. It has implemented and freely distributed grid middleware tools for security, resource management, information handling and data transfer.

- Grid Security Infrastructure (GSI) [24] for establishing a grid user's identity. As an implementation of the security protocol based on public key cryptosystems [85], Globus GSI supports services such as single sign-on (so that a user need only log in once to a grid), and credential delegation and mapping. With these, a user's access privileges may be delegated to processes and grid-wide user identities are associated with resource-specific identities.
- Globus Resource Allocation Manager (GRAM) for grid-wide resource management [17][18]. The GRAM protocol allows the users to securely make job submissions and pass jobs to the appropriate local scheduler (or job launcher) for execution. GRAM employs a custom Resource Specification Language (RSL) [88] based on XML [61] to express job requests.
- Monitoring and discovery services (MDS) [16] to derive and provide job and grid-related information. MDS enables the user to obtain details of available hardware and software resources, and to determine their status and availability. This tool is based on the Lightweight Directory Access Protocol (LDAP) [74] for saving, managing and requesting information.
- GridFTP [2] for distributed data access across heterogeneous date resources [15]. Extensions to the standard File Transfer Protocol [62] have been implemented for high-performance data access in grids. Globus Access to Secondary Storage (GASS) [5] tools also provide data transfer with transparent caching.

The Globus Toolkit has undergone several changes since its first release. One of the most significant of these was the move towards a service-based instantiation, based upon the Open Grid Service Infrastructure (OGSI) developed by the Global Grid Forum [83]. These changes were reflected in Globus Toolkit v3, or GT3, which was released in July 2003. The latest changes evolve the Globus Toolkit toward the Web Services Resource Framework (WSRF) [82]. The WSRF proposal brings OGSI concepts more into line with Web Services. Globus Toolkit v4, or GT4, is slated to appear sometime in early 2005 and will introduce support for WSRF. Although recent revision and refining of basic grid protocols and standards by the Global Grid Forum [63] aim to define grid services that adhere to web-service specifications, the functionality provided by the Globus tools remains essential in a grid environment and continues to be provided in new releases of Globus.

A. The UH Campus Grid

Several major research activities at UH, including AQF, require access to considerable computational power and, in some cases, large amounts of storage. To accommodate many of these needs locally, a decision was made to create and operate a campus-wide grid that combines central resources at the university's HPC Center (UHHPC)

with departmental clusters. In addition to providing an infrastructure for launching the various components of AQF runs, other applications in such disciplines as chemistry, physics and engineering can be fully and securely integrated into the same grid architecture, resulting in a computational environment with a richer set of resources than otherwise locally possible.

The campus grid currently consists of a heterogeneous cluster of Sun SMPs, a Beowulf cluster and an SGI visualization system, with 9 TB storage, at UHHPC, a cluster of Sun SMPs in Computer Science and several Sun workstations and Linux clusters operated by the Math Department and the Geophysics Department. The AQF effort has access to two additional Myrinet-based 64 node Intel clusters and 5 TB storage for experimentation. This grid is available to a broad cross-section of faculty and is deployed for both research and teaching. The facility is heavily utilized, with high average waiting times in the queue for submitted jobs.

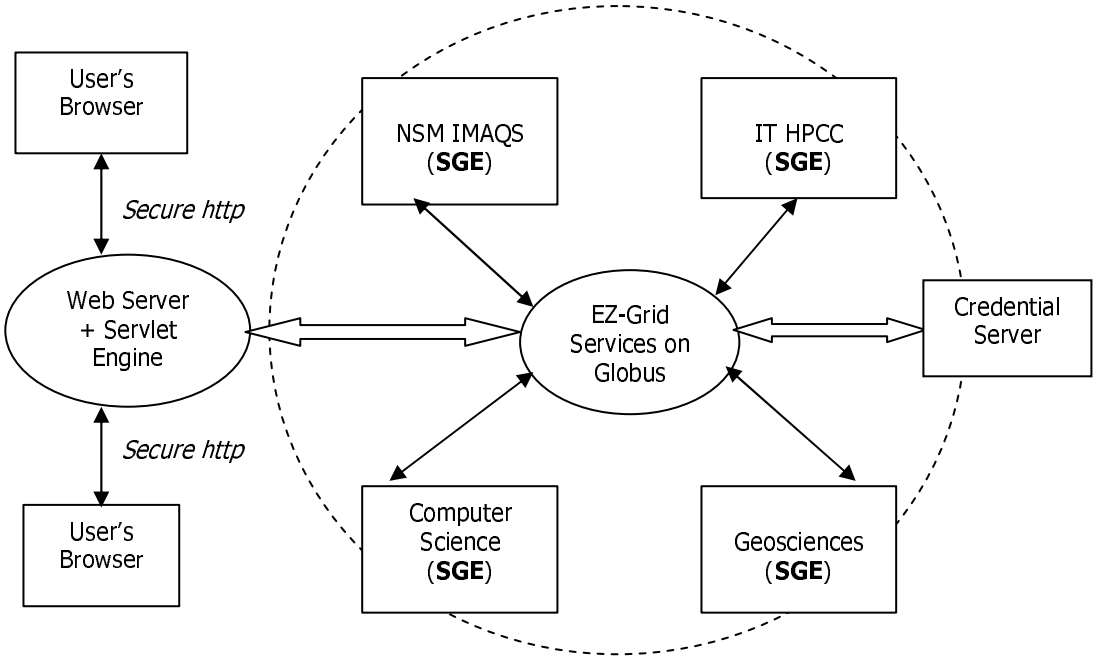

Fig. 3. UH Campus Grid's Initial Setup

Each resource in the campus grid is locally administered. Sun Grid Engine (SGE) has been installed to manage the resources within the individual administrative domains. Inter-departmental security is enabled via the certificate-based authentication mechanisms provided by the Globus toolkit, according to which the identities of users and resource are verified based on their public keys and the associated certificates. Certificates must be issued by a trusted certificate authority. UHHPC serves as the certificate authority in our campus grid and is also responsible for granting grid accounts to faculty and researchers across campus. Individual departmental resources are configured to accept only these certificates in order to protect them from unauthorized access by non-accredited users. The initial configuration of the UH campus grid is illustrated in Fig 3.

B. EZ-Grid Portal for UH Campus Grid Services

It is a daunting task for many application scientists to interact with grids using the interfaces supplied by Globus. One goal in the design of our campus grid environment is to make it as easy as possible for users to interact with the grid services provided. EZ-Grid [14] is an on-going project that focuses on making it easier and more efficient for application scientists to use grids. EZ-Grid is a light-weight, freely available implementation of a web-based portal for ubiquitous access to grid functionalities. The software is very small in size and exhibits minimal external software dependencies, while providing a convenient interface to all functionalities of the Globus toolkit, including security, resource information, data management and job submission services. The Globus Java CoG Kit [32][33] has made it relatively easy for us to access Globus functionality. The portal classes consist of a set of portal services, implemented as Java Servlets [71], and a credential server; they can execute on any web server that supports Java Servlets. The system architecture, showing the relationship of EZ-Grid to other middleware, is shown in Fig 4.

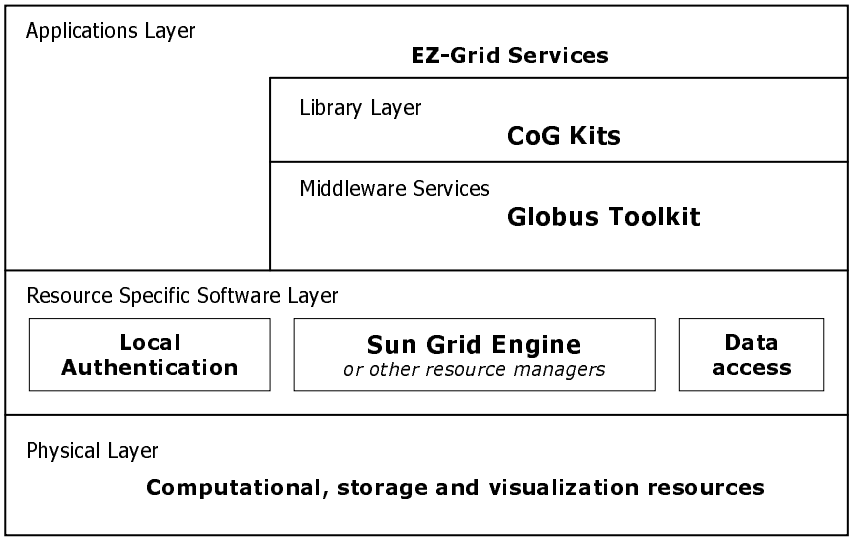

Fig. 4. EZ-Grid Software Architecture

The initial version of EZ-Grid provided the following major functions to a grid user:

- Grid authentication: Globus proxy creation and management using GSI and X.509 certificates. This allows the user to seamlessly establish his or her identity across all campus grid resources and mutually authenticate with them.
- Resource and operational information: Viewable information on status of grid resources, with static attributes such as operating system version and CPU count, and dynamic attributes such as CPU loads and current length of job queue. Users can check queue availability, or the status of their submitted jobs. Additional EZ-Grid specific information includes application profiles (metadata about user applications) and job execution histories.
- Job specification and submission: the GUI enables the user to specify a job and its requirements, including the resources needed for its execution, and to supply information to the target resource management system. Automated translation of these requirements into RSL and subsequent job submission via GRAM are supported by the portal.
- Job management: Storage and retrieval of relevant application profile information, history of job executions and related information. Application profiles are metadata to characterize applications that can be composed by the user.
- Data handling: Users can transparently authenticate with and browse remote file systems of the grid resources. Data can be securely transferred between grid resources using the GSI-enabled data transport services such as Grid FTP.

In order to complement the static resource information that is standard in a Globus-based environment with queue configuration information and dynamic queue status, interfaces were developed with the local resource management systems. Our environment primarily uses Sun Grid Engine (SGE) [91] to manage the workload; it provides a relatively straightforward set of commands that enable us to derive the required information. Users can thus check the status of their jobs, the load on the individual resources and queue availability. Additional information provided includes application profiles (metadata about applications) and user-specific job execution histories.

IV. Requirements for a Grid-enabled AQF

Our campus grid and EZ-Grid portal software were tested as a part of student coursework [64] and by graduate students engaged in computational science throughout the university. It has been used for the execution of a variety of jobs. User feedback helped improve the software, and in the course of its deployment, we have gained valuable experience of grid challenges facing both users and administrators. However, until now EZ-Grid has been used to run jobs consisting of a single executable. In order to support the AQF project's goal of producing reliable, timely and accurate air quality results using resources across our campus grid, more work was needed. Such a complex

application poses a number of additional challenges in terms of both the functionality and reliability of our campus grid environment.

The AQF application requires the execution of multiple limited area weather models on increasingly smaller, but more closely meshed, domains in conjunction with the chemical model. It also relies on global weather data that is automatically retrieved each day. In order to support execution of the entire application, a computational grid environment must enable the specification of the complete job including the interactions between its various components; it must allow for the automated retrieval of global weather data and subsequent initiation of preprocessing, must start the weather model once the initial data set is ready, and be able to launch other executables when the corresponding input data has been produced, according to the application cycle previously described.

Software is thus needed to enable a complete description of the application and its execution cycle, to regulate its use (including such things as determining which users may configure aspects of the different codes), for scheduling the individual executables, for transparently transferring files between machines according to the workflow, and for monitoring the state of the on-going computation and responding to performance (and other operational) problems.

In the course of this requirements analysis, we collaborated closely with the AQF application scientists in order to better understand their specific needs and ensure that our revisions meet them, as well as with system administrations in order to understand the challenges involved in maintaining grids and dealing with different kinds of users. We further investigated the major impediments to grid-enabling our AQF applications. The outcome was a refined set of requirements and a plan for the development of a more comprehensive environment to support the AQF project lifecycle.

A. Workflow Requirements

During the course of execution of the AQF application, terabytes of data (meteorological data, emissions inventory data, air quality monitoring data, and air quality simulation output) need to be transferred among the modeling subsystems that may run on different computer platforms (parallel or sequential) at different physical locations (on campus or state-wide). Even within each subsystem, a job will be accomplished via the cooperation of different modules, potentially executing on different machines. For example, the MM5 weather forecasting system provides meteorological data for the Air Quality Modeling tools. But in order to prepare the data for input to the AQF subsystem, the input data for MM5 itself must be obtained, preprocessed to create the initial conditions, the simulation performed and post-processing tasks carried out. This is a cumbersome process that typically requires the use of scripting languages or human control.

AQF tasks include running MPI jobs on hundreds of processors, transferring terabyte-sized files to visualization servers at different sites, and archiving large data sets to mass storage. Support is needed to graphically define a complex sequence of tasks that couple forecasting, emissions processing, and chemical transport simulation. AQF team members require a graphical environment for specification of the workflow of a job being submitted, as well as a workflow language that permits expression of the job's dependencies for co-scheduling.

Workflow systems are needed to manage complex jobs such as this that consist of multiple executables and dependencies between them. Without this, AQF execution is managed using shell scripts that are platform-dependent and do not provide mechanisms for data communication between dependent activities. Challenges to support for workflow in a grid environment include the existence of multiple administrative domains and the dynamic nature of grid resources. Scientific applications with complex workflow such as AQF may have needs that are specific to the scientific domain. Moreover, large-scale codes typically require at least some of the following:

- Support for very large datasets
- Support for adaptation to changes in the environment (especially where resources can dynamically change)
- Hierarchical execution, with sub-workflows created and destroyed when necessary
- Execution of large number of jobs with varying parameters
- Monitoring and dynamic control of workflow execution

B. Requirements for Grid Metascheduling

Grid metascheduling is the process of making scheduling decisions involving resources over multiple administrative domains. One of the primary differences between a grid scheduler and a local resource scheduler is that the Grid scheduler does not own the local resources and therefore does not have control over them. The grid

scheduler must make best-effort decisions and then submit jobs to the resources selected, generally acting as if it were the user. Furthermore, the grid scheduler does not have control over all of the jobs submitted to the local resource scheduler, so decisions that trade off one job's access for another's are very hard to make in the global sense. This lack of ownership and control is the source of many of the problems that need to be solved in this area. For the AQF application, a job consists of several dependent tasks and placing these tasks on the appropriate resources across a grid is much more complex than scheduling a single-executable job. Also the scheduling decision must ensure system-wide Quality-of-Service: in our case, this means we must make sure the prediction results are generated in a timely fashion. In addition to the general challenges of metascheduling, specific requirements for AQF metascheduling include:

- Workflow orchestration: A submitted AQF job is not a single task or executable that can be scheduled to one single resource, but includes tasks that depend on each other in a complex fashion, and also on the file transfer between grid resources. The metascheduler cannot schedule the tasks one by one onto resources once the dependencies are resolved. Instead, the metascheduler should make a one-time assignment of tasks to resources and possibly ask for advance reservation from the local resource manager for some tasks. When launching each task, the metascheduling system also has to make sure that the workflow order of AQF tasks is correctly maintained.
- Time constraints: The AQF application has a recurrent, i.e., typical, mode of operation and thus identifiable typical requirements. Each day in our current development mode, AQF starts execution at 15:30p.m. and completes around 7:00a.m. the next day. The operational mode will require much faster completion time but will also be predictable in terms of required start and end times. The metascheduling must guarantee these. When a task is submitted by the metascheduler to the local resource manager, the metascheduler relinquishes control of the job and the job may potentially be held in the queue of the local resource manager for a long period of time before being launched. Metascheduling solutions should be carefully designed with this in mind, and wherever possible, utilize deadline or priority scheduling options that ensure a job's timely initiation.
- Data handling issues: AQF applications download and generate large amounts of data each day. Proper handling of this data is necessary if an AQF job is to be successfully distributed across grid resources for execution. Efficient data movement between resources, without unnecessary waits, strategies and tools for management and archival of this large and growing set of data, must all be provided.

C. Grid Security Requirements

It is hard to meet all security needs in a grid while remaining transparent for the user. General problems of ensuring system integrity in a networked environment must be dealt with; data communicated between grid resources must not be tampered with. The relatively straightforward task of logging in to a resource is complicated by the need for generality and transparency of authentication. Grid applications such as AQF may spawn processes that communicate with one another across multiple administrative domains: to clearly associate all resource requests and utilization with an accredited user, a single grid-wide identity is needed, along with a means for deriving local resource-specific user identities, and corresponding access rights, from that grid identity. Since grid jobs may dynamically acquire new resources, for instance to satisfy performance faults, grid authentication strategies must also provide for the delegation of rights to a remote process. To protect against their potential abuse, they should expire after an associated time period. Any resource request made on behalf of the user after this time period thus requires a fresh delegation. Solutions must allow for a variety of different site policies with regard to security, and need to interoperate with a variety of pre-existing local authentication schemes.

The Globus toolkit's grid security infrastructure is almost universally used to provide authentication in grid environments, including ours (cf. Section III-A). It provides the services described above. Users mutually authenticate with resources on the grid and securely spawn and manage remote computations using X.509 certificates. GSI is primarily based on public key infrastructure (or asymmetric key) cryptography (PKI) and uses Generic Security Services API (GSS-API) [34]. Interoperability with other technologies such as Kerberos [41] has been addressed (e.g. KX509/KCA [73]).

Note that there is an inherent problem for scheduling that arises as a result of the way in which user identities are managed. The strategy for associating grid-wide and local identities does not expose information associated with the grid-wide identity to local resource management systems. These local systems are responsible for optimizing

the utilization of resources under their control, for which they may exploit resource usage privileges and scheduling strategies associated with local user accounts. We need to leverage their optimization capabilities for grid application execution, but as a result of their limited information they cannot take the wider environment into account.

Although it is probably the most widely deployed part of Globus, our initial experiences using GSI grid revealed several different kinds of weaknesses with this solution. We describe our security infrastructure briefly before describing the problems identified.

In our setup, user authentication with EZ-Grid was equivalent to performing a grid proxy initiation with the Globus toolkit through the portal. Our campus grid users obtained certificates from UHHPC for authentication with the portal as well as with the grid resources. A credential server acted as a secure repository for the user credentials (X.509 certificate and key pairs) and the proxies associated with the grid-wide user identity. This realization permitted unlimited user mobility through a browser and enabled secure storage and export of new or renewed certificates and keys to the credential server through the portal. Alternatively, a MyProxy [39] server could have acted as an online repository for user proxies.

The first shortcoming of this approach is the relatively cumbersome setting up process, which involves a number of steps to be performed by both the end user and system administrator. The following list indicates some, though not all, of the major setup steps required by GSI.

1) The user has to generate certificate requests with the appropriate fields in the DN
2) User certificates must be signed by a CA acceptable to the target resource
3) The user sets file permissions appropriately for the credentials before signing-on
4) The user conveys his/her DN in the certificate to the administrator of the remote resource
5) Each remote administrator records the user's DN and the associated mapping to the local resource-specific user
6) The user configures local settings to accept the credentials of the CA that issued remote host certificates and the credentials of the remote resource itself
7) The path must be set correctly for the trusted CA certificates to enable subsequent mutual authentication with the target resource
8) The user creates the proxy with the appropriate credentials and authenticates with the resource

The steps 4 through 7 are repeated whenever the user gains access to a new resource that trusts the CA that issued the user's certificate. Steps 1 through 7 are repeated whenever the user gains access to a new resource that requires user credentials issued from a different CA.

The relative complexity of this process was the cause of frequent confusion and error among grid users, who were often unfamiliar with PKI. As illustrated by a simple search on Globus mail archives, numerous deployers have also encountered problems in setting up GSI correctly. This problem is compounded by the complexity involved in establishing and using PKI itself. Next, we explain the shortcomings associated with how we set up the portal authentication.

Our portal's authentication setup had obvious shortcomings in terms of maintenance and in accommodating an increasing number of users and resources. There are two distinct authentication tasks involved. First, the user has to prove his identity to the portal server while logging in; second, the user does mutual authentication with the grid resources that he wants to use. In our initial setup, both these tasks used GSI and the same PKI credentials. Lack of separate mechanisms for these two tasks hindered the flexibility of portal usage and posed increased security risks if credentials were compromised. The portal credentials were to be refreshed whenever the user's grid credentials expired and vice versa. Portal authentication setup could not be changed independent of the grid credentials.

This setup posed significant burden to the system administrators in operating the portal. A variety of management tasks required system administrator effort that would not be possible with large numbers of users or resources. In particular:

- Whenever a new resource was added to the grid, GSI-specific configuration tasks had to be manually carried out to ensure correct identification between local and global user identities, a non-trivial effort.
- The removal of a user's grid account required manual editing of configuration files for all resources that the user had access to. In the case of compromised user credentials, this would potentially require editing of all such files, a serious problem if the number of participating resources is high. Similarly, any revocation of a user's certificate for other reasons requires the manual updating of the configuration files to ensure that future accesses via the revoked credentials are denied.

- If users may have different identities/roles, potentially with distinct access rights and priorities, they need multiple sets of credentials. It proved to be difficult to decide which credential to use for the initial authentication. There was also a problem translating between the various credentials when the user required access to multiple resources that did not accept a single CA's credentials.
- Although it was a relatively easy task to make software such as ftp and ssh available through the portal, whenever HPCC wanted to provide new services (e.g., user training materials) to campus grid users, they had to rely upon a different authentication scheme for establishing identities. Or, they had to rely upon the GSI mechanism even for new non-grid services.

Improvements were clearly needed to make the security infrastructure easier to deploy and maintain for both users and system administrators.

V. UH Production-Quality Grid for AQF Application

To support reliable, timely and accurate air quality prediction using our AQF applications, our UH campus grid and its supporting software are evolving into a production-quality environment. In order to add required new functionality, we have carried out an extensive study of related systems to identify their strengths and shortcomings. We leverage and extend state-of-the-art grid technologies to develop our AQF-specific grid middleware. In the coming sections, we explain our strategies for supporting the complex AQF workflow, metascheduling across organizational domains and seamless grid authentication.

A. EZ-Grid Workflow Support

Grids encourage the deployment of applications with non-trivial workflow and in consequence, several workflow orchestration tools have already been developed, or are under development, by the Grid community, and relevant standards have been defined or are under way. BPEL4WS [52] is a popular business language for scripting workflow to integrate multiple services to jointly accomplish a complex task. A workflow engine acts as the agent that follows the BPEL4WS specification document and contacts each of the services required by the specification following the order specified. BPEL4WS is constantly being improved. However it is closed source and was not designed for long running workflows. Consequently, the Grid Services Flow Language (GSFL) [46] has been defined to enable the description of workflow for Grid services within the OGSA framework.

Condor is probably the best known workflow system for scientific computing. The Condor DAGMan [58] allows dependencies to be expressed between Condor [35] jobs (executables) and supports the submission of multiple jobs to Condor in the proper order. For DAGMan, a configuration file created prior to job submission describes the workflow in form of an acyclic graph. In it, the jobs are represented as nodes of the graph, and the edges identify the dependencies between them. Priority relationships between input, output, and execution of the corresponding programs can thus be specified. However, this software can only be used for Condor jobs and in order to interface with other grid codes, a separate abstraction layer is needed. An alternative to the approach adopted by Condor is the Directed Cyclic Graph (DCG) used by Triana [94] as its method for modeling the workflow process. Triana is built on the idea of composing applications from reusable components, as are a number of other systems including WebFlow [6] and Symphony [36]. WebFlow provides a web-based visual programming environment for high performance computing software. Symphony and Triana offer similar functionality for the development of distributed computing software from predefined software modules, whereas Common Component Architecture (CCA) [54] focuses on composition of software components at runtime.

However, most of these workflow systems are designed to be used by experts and are not suitable for use by novice users. Gridant [4] and Karajan [72] are efforts that target the user without expertise in sophisticated workflow systems. Both aim to provide a convenient tool to the Grid community that allows the expression and control of the execution sequence in a workflow computation. As parts of the Globus Commodity Grid kit (CoG) [32] [33], both Gridant and Karajan have the ability to submit jobs to multiple resources managed by GT2, GT3 and the future GT4. Gridant is a simple client-side workflow specification system that makes use of Apache Ant [50] [51], a popular build tool extensively used in the Java community, as its workflow engine. The functionality of Apache Ant is extended by Ant tasks, which are specifically suited for the grid environment.

Karajan, which provides workflow specification and a workflow engine, enables the definition of complex jobs for execution in a computational grid and offers advanced features such as failure handling, checkpointing, dynamic

workflows, and distributed workflows. A workflow can be classified as centralized or distributed according to the manner in which tasks are distributed within a workflow. A workflow in which all tasks are executed at one location is called as centralized workflow. In distributed workflow tasks can be executed at multiple locations. Workflows in Karajan are defined using a language based on XML; it is extensible through Java. The execution engine in Karajan is based on an event handler. Karajan defines elements which are building blocks for functionalities such as parallel processing, parallel iterators, and Grid elements such as job submission and file transfer. Elements react to events received from other elements and generate their own events.

After evaluating existing technologies, including the above, we decided to leverage the Karajan software to provide workflow support in our environment. Karajan is open source and can easily be modified to suit AQF needs. Karajan supports sequential and parallel execution containers that allow subtasks to be executed in sequence or concurrently, as desired. Tasks and dependencies between them can be conveniently expressed. Inter-task dependencies may be specified as constraints on the occurrence or the temporal ordering of significant events generated by the involved tasks.

Yet, one major weakness of Karajan in supporting AQF is the lack of support for metascheduling and it only submits the tasks of a workflow job one by one. Thus we must extend some of the Karajan workflow component to integrate with our metascheduler. Metascheduler is responsible for assigning individual tasks to different resources and for enforcing task dependencies (this procedure is described below). The scheduler also coordinates the execution of tasks in the workflow. In [43] a distinction is made among three scheduling approaches: centralized (a single scheduler schedules the tasks of all concurrent workflows), partially distributed (a local scheduler for each workflow), and fully distributed (no scheduler is used, but task agents coordinate their execution by communicating with each other). We have adopted the approach of a centralized scheduler. Overall, the workflow is managed by a Grid metascheduler, which assigns and schedules jobs to different machines, while the individual tasks are managed by local schedulers.

We also aim to provide the option of fully automatic scheduling of AQF workflow jobs based on our profile data and experimentation with a variety of execution scenarios. Current research in our team also considers automated resource brokerage. This is highly desirable to automatically and transparently select suitable resources to test environment model configurations and find optimal setups for different geographic and seasonal conditions.

In addition, GUI support from Karajan is very simple and not integrated with web portal. In our work, a high-level GUI will be provided to help a user define complex tasks (workflows) using the extended Karajan and will be integrated with the EZ-Grid portal environment. Additional on-going work addresses features that allows users to store and schedule workflows when desired. These workflows can then be configured to automatically run, either periodically or at a specified time.

B. QoS-Guaranteed Metascheduling with Workflow Orchestration

As part of our efforts to develop a metascheduler to support AQF project, we have tested a number of related software systems to determine the one most appropriate for our needs. Global metaschedulers are relatively new in the field of grid computing. Existing global schedulers such as Maui and CSF are sophisticated, but unfortunately cannot fulfill our needs. The Maui scheduler is not OGSA-compliant, has no workflow support and is also not open source. The Community Scheduler Framework (CSF) [86] is a global scheduler that interfaces with local schedulers such as OpenPBS [84], LSF [87] and SGE. However, CSF currently has no support for specifying workflow. Other essential features missing from existing global schedulers are data-aware scheduling and interfaces with portals. Data-aware scheduling refers to an ability to select the compute resource that is closest to the data. This is likely to be an important feature for applications such as AQF, which transfer large sets of data.

The task of the metascheduler is to select appropriate resources for submitted jobs, according to the current or predicted resource usage information and user job information. It also helps to set up the file transfer channel that will be used for moving data between tasks or jobs. However, the process of choosing appropriate resources is very complicated. The metascheduling 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. In our initial implementation, we divide this process into two stages; first, we select resources based on the job specification and static resource information, such as the number of CPUs, memory sizes, etc. Secondly, we order the selected resources based on dynamic resource information, which includes resource usage value (RUV) and job average

waiting time (JAWT) on specific resources. The first step performs the simple match-making of each attribute of the job specification with known resource information. For the second step, RUV and JAWT are combined into a single normalized value, Resource Load Value (RLV) and based on this RLV, the metascheduler will order the selected resources and choose the ”right” one for the user’s job.

The current metascheduling system is designed to find suitable resources from a grid for a single job. Scheduling dependent tasks of a job could be simply implemented on top of single-job metascheduling by scheduling each task when its dependent tasks are completed. But a major problem of this approach is that the metascheduler cannot in general guarantee the scheduled task will be launched right after it is submitted to the local scheduler and it may wait in the local queue for some time. Our approach makes a one-time choice that assigns resources for all the tasks, and then the metascheduler submits them to the local scheduler. The required resources of the submitted tasks will be held or reserved by the local scheduler during their execution time. The metascheduler predicts this time based on available information on the duration of each task and the amount of data produced. The high predictability of AQF runs, which generally use the same mesh sizes and number of time-steps, will enable us to do so accurately. We have extensively profiled the complete application on the various grid platforms and configurations at our disposal in order to better understand its likely runtime behavior and to determine grid configurations that are useful for execution. This may need to be partially repeated when systems are extended or upgraded.

Accurate run-time resource information is also essential for the metascheduler to make the best decision globally. The metascheduler can exploit both static and dynamic information in the two stages of match-matching. First, static information that captures resource setup details such as platform, OS, library installed, etc, will be used in the first-stage filtering by metascheduler. Static information is provided by Globus MDS or configured in the information services by system administrators. Second, dynamic information that captures the runtime behavior of grid resources, such as current load, queue information, available memory and swap, available network bandwidth, and so on, will be used in the second stage of scheduling. This information is provided by either the local resource manager or third party resource monitoring tools. Since our AQF jobs will run daily at the same time, resource usage forecasting may be employed to guide the metascheduler in resource assignment.

1) Workflow Orchestration: To integrate Karajan with our metascheduler, we extend Karajan’s workflow descriptions with two sets of information necessary for global scheduling: first, task dependency details, which we call dependency elements; and second, the profiled task execution detail. Task dependencies are represented by DAGs like those in Condor. Dependency elements, such as file name, parameters passed, are appended to the DAG edges, which thus have richer information than just the relationship. The DAG vertices, which represent tasks of a workflow job, are linked with the execution details, such as the execution time on various number of CPUs on different grid resources.

Using the extended workflow description, the metascheduler assigns tasks to resources and submits them accordingly to the local scheduler. To do so, the metascheduler first predicts when each task can start by traversing the DAG; it then checks resource availability for each task at its starting time. During this calculation, the time needed to handle dependency elements may also be considered, one typical example of which is file transfer between tasks. This is further discussed in Section V-B.3. To mitigate the effect of inaccurate prediction, the metascheduler assigns buffer time or a grace period for each task.

Although the tasks are submitted to the local scheduler once they are assigned to resources by the metascheduler, they will not be launched until their dependencies have been resolved. This is accomplished via the sending of corresponding events by the dependent tasks when they complete. Until these are received, the tasks are held in the local scheduler’s queue. At that point, the event handler will release the task from the queue and it will be dispatched to the compute nodes. The metascheduler also receives the event notification and triggering information to enable it to track the current flow of task executions. It can use this to adjust the start time of other unfinished tasks. If the metascheduler does not receive the event notification after the established grace period, it will check the current status of their dependent tasks. If they are still healthy, it may do nothing but monitor it more frequently. In case of severe delays or the failure of tasks, the metascheduler may force the local scheduler to launch or dynamically modify the assignment of resources to uncompleted tasks. The failure handling also causes the metascheduler to adjust the event chains established previously.

2) Time Constraints: If not properly scheduled, an AQF task may be held in the queue of a local resource manager for a long time and this is not controlled by the metascheduler. Our ability to schedule executables to reliably

produce results at the required time hinges on the ability of the local schedulers to perform advance reservation and backfilling. The high predictability of AQF jobs permits us to reserve resources or create temporary dedicated queues for AQF tasks in order to reduce or avoid waiting time in local queues. We have tested these capabilities in SGE, LSF and PBS Pro, all of which provide resource reservation features. We are currently developing a uniform interface to enable our metascheduler to make best use of these features. This interface complements the DRMAA [59] interface to local schedulers. DRMAA is currently supported by SGE and Condor and provides job control interfaces. We plan to submit our interface for resource advanced reservation to GGF.

Another key feature of our metascheduling approach that helps to address the time constraint issue is the immediate submission of tasks with their predicted starting time when the resource selection decision is made. The local resource manager will hold the resources for the task if its the dependencies are not resolved at the specified start time, or backfill to permit other jobs to run when the AQF task is waiting. Preemptive scheduling may be employed to ensure that the high-priority AQF tasks can use the resources if they are approaching their deadline.

3) Data-aware scheduling: If the execution is to span several grid resources that do not have shared file systems efficient data handling is essential. While file-staging in the local resource manager or Globus GASS can be used to transfer files for an individual AQF task, they also introduce several problems. First, both are completely out of the control of the metascheduler once the task is submitted. Further, there is a higher probability of failure when transferring large files than with small file transfer; any failure should be detected easily and quickly to initiate re-transfer. If a task is stopped because of the failure of file staging or a GASS transfer, the metascheduler would have to identify the details of the job error report from local resource manager to discover the root cause, and resubmit the tasks with the any required corrections. This process would be very inefficient, would complicate the design of the metascheduler unnecessarily and would probably pollute the scheduling decisions already made. Moreover, time spent in file staging or GASS is normally ignored by a scheduling system. But our AQF application requires large file transfer and has time constraints, so that this time should be properly allocated by the metascheduler when assigning AQF tasks to grid resources. Thus we have rejected the use of these mechanisms and devolve this responsibility to our metascheduler.

In our metascheduling strategy, we consider large file transfers to be separate tasks in the workflow. The approximate time for the transfer is calculated from the predicted file size and network bandwidth between the source and destination hosts obtained via monitoring. Also, in the process of assigning resources to tasks, metascheduling will consider the existence of shared file systems within or between resources, and thus eliminate unnecessary transfer costs. The design of our metascheduler recognizes and deals with situations when output is not produced in a timely fashion by any one of the executables in a job. This is based upon the profile data and active monitoring of job status.

C. Revised Sign-On and Authentication Setup in UH Grid

As a result of our experiences with GSI, we searched for alternative approaches to realize a flexible and scalable approach to grid authentication. Our portal authentication phase had to be decoupled from the grid authentication and hence made independent of GSI. We required a new web-based authentication mechanism to achieve this.

CoSign [56] is an open source project at the University of Michigan to provide a web-based authentication system. Users need to authenticate only once per session in order to access any number of Cosign-protected services. It is based on the idea of using a central server for authentication. Cosign provides a much simpler web-based initial logon scheme than does GSI and permits use of existing authentication schemes such as LDAP, Kerberos, username-password and X.509 certificates. In a CoSign-based security scheme, our portal services (including those that let the user access grid resources) would check with a central CoSign service to ensure that a user is logged-in. This approach improves portability and reduces administration overhead. Using a central server for authentication also tremendously simplifies the configuration needed to support user certificate-based authentication (e.g. KX-509) across campus web services. Unlike other cookie-based web authentication solutions, Cosign does not employ domain or otherwise public cookies to allow cross-server authentication. The Cosign server has its own cookie as do the departmental web servers. The Cosign cookie is available only to the Cosign server; the departmental service cookies are available only to the departmental servers. Cosign leverages its central state database to allow simple, effective user-initiated logout at the end of a session.

We have revised our UH campus grid so that it relies upon a centralized CoSign authentication server for portal authentication. Once the user identity has been successfully established through this central server, the user can access any number of services protected by the Cosign server, including grid services. Thus, user access to grid services is considerably simplified in comparison to our earlier setup. The portal authentication phase relies solely upon CoSign and is completely independent of grid authentication credentials and GSI. Further, the two authentication mechanisms (CoSign and GSI) can be setup and configured independent of each other, providing ease-of-use for administrators. A stand-alone grid proxy creation service now allows the user to choose among multiple GSI credentials to perform grid authentication. Access to this service, in turn, is protected by the central CoSign server. Now, the user has a clear separation between authenticating to the portal and to the grid resources.

It is now a simple task to add new services to the portal, which entails protecting them by the central Cosign server. All services check for Cosign authentication and user identity before granting access to the user. The Cosign server can be replicated in order to achieve better load balancing and avoid a single point of contention and failure. However, adding new grid resources still poses a problem as it requires that the GSI-related configuration be repeated on each target resource. This is a concern we plan to address in our future work.

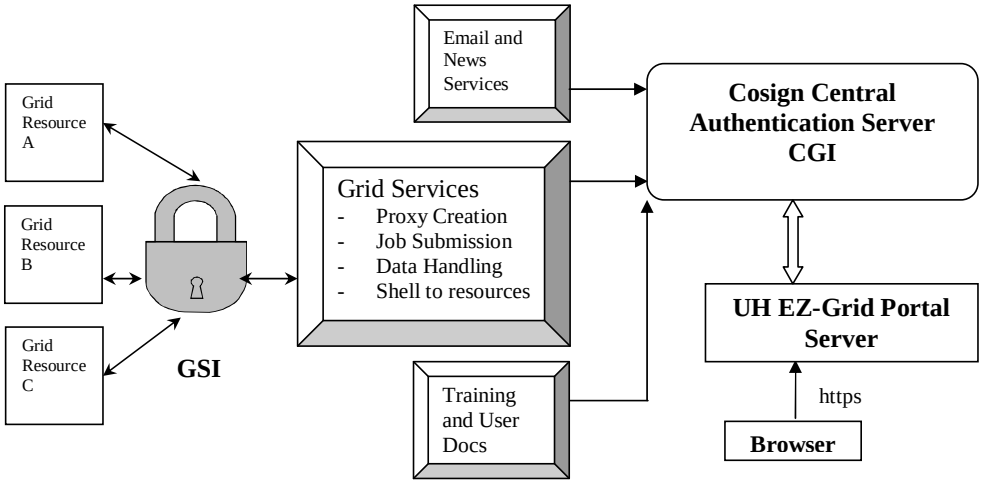

Fig. 5. CoSign-based UH Campus Grid Authentication Setup

Under the revised setup, users authenticate to the central CoSign server using a simple username/password combination. The success of this process is indicated by the establishment of a login cookie for the user on the central server. Then whenever a user accesses a service (protected by CoSign) a service cookie is established. The central server orchestrates this process by associating the service cookie with the user's login cookie. The user can simply access CoSign-protected grid services for proxy creation (this requires the correct passphrase entry however), job submission and other related grid operations. The user can also logout by simply clicking a URL that clears the login cookie, after which subsequent operations require fresh authentication with the central CoSign server. The process is illustrated in Fig 5.

It is important to note that our revised portal only simplifies the user tasks of signing-on and accessing grid services. It does not address the server-side work required to maintain correct credential configurations and identity mappings. We have yet to deal with a scenario where the user acquires access to resources that require a new set of credentials, potentially because they expect certificates from a different CA.

VI. Conclusions and Future Work

Grid environments enable the creation of new infrastructures as ”virtual organizations” that seamlessly aggregate distributed and dynamic collections of resources. Campus grids provide an opportunity to help researchers solve computationally demanding problems in science and engineering. However, simplified access to grid services is essential if computational scientists are to fully utilize them. Grid portals can help raise the level of user-grid interaction. They enable web-based access to grid resources and services and 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. This portal is being extended and adapted to meet the specific needs of the AQF project. The AQF application consists of a number of executables that interact via the transfer of relatively large files; it has a highly predictable execution behavior. New functionality will enable AQF researchers to fully automate the process of specifying and starting their complex jobs that may utilize a variety of hardware resources on our campus.

Major improvements include support for specifying, scheduling and executing workflow jobs that entail the transfer of large data files between executables. They also include support for improved user authentication. To achieve our goals, we have evaluated a variety of existing approaches and where possible, exploit existing implementations. Our current workflow system allows users to conveniently and flexibly combine different computational tasks within a module e.g MM5.

Existing metaschedulers make resource assignment decisions mostly for single-executable jobs and do not have any QoS features. Our scheduler integrates with and extends Karajan workflow systems to support AQF jobs with complex dependencies, and addresses AQF-specific issue of time constraints for better QoS. To improve the decision making, the scheduling process exploits additional information including job profile data. Time cost for data transfer is considered also to make sure that the coordination of dependencies are properly handled and incur minimum unnecessary delays in the coordination. We are also prototyping network-bandwidth- and topology-aware scheduling in the metascheduler. This is especially useful for our AQF campus grid where huge datasets need to be downloaded and transferred before and after the execution of each AQF module.

The need for users to deal explicitly with certificates proved to be one of the biggest hurdles in user-grid interaction. We modified our campus grid to use the CoSign software to simplify this process, although it did not alleviate the system administration burden. We need to continue to search for ways to simplify all aspects of grid authentication. Since GSI is the de-facto authentication mechanism, a solution should ultimately use the GSI fabric; however, it is important to make it easier to achieve grid security from the perspective of both users and system administrators. We plan to study information retrieval algorithms [48] for key management and retrieval based on search query formation. Also, key striping techniques might be leveraged for increased fault tolerance and protection against compromised keys.

ACKNOWLEDGMENTS

We wish to thank Dr. Daewon Byun for his support for our efforts to explore the AQF application and Dr. Jiwen He for his help in understanding MM5 and its deployment. We would also like to thank the HPCC team for their help in the construction of a testbed to facilitate this initiative as well as for their feedback on our efforts.

REFERENCES

[1] 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.

[2] 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*, Parallel Computing 2001.

[3] G. ALLEN, D. ANGULO, I. FOSTER, G. LANFERMANN, C. LIU, T. RADKE, E. SEIDEL, J. SHALF, *The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment*, International Journal of High-Performance Computing Applications, Volume 15, Number 4, 2001

[4] K. AMIN AND GREGOR VON LASZEWSKI, *GridAnt: A Grid Workflow System. Manual*, February 2003

[5] J. BESTER, I. FOSTER, C. KESSELMAN, J. TEDESCO, S. TUECKE, *GASS: A Data Movement and Access Service for Wide Area Computing Systems*, Sixth Workshop on I/O in Parallel and Distributed Systems, May 5, 1999.

[6] D. BHATIA, V. BURZEVSKI, M. CAMUSEVA, G. FOX, W. FURMANSKI, AND G. PREMCHANDRAN, *WebFlow A Visual Programming Paradigm for Web/Java Based Coarse Grain Distributed Computing*, Concurrency: Practice and Experience, vol. 9, no. 6, pp. 555577, 1997.

[7] R. BUTLER, D. ENGERT, I. FOSTER, C. KESSELMAN, S. TUECKE, J. VOLMER, V. WELCH, *A National-Scale Authentication Infrastructure*, IEEE Computer, 2000.

[8] D. W. BYUN, J.K.S. CHING, *ed., Science Algorithms of the EPA Models-3 Community Multi-scale Air Quality (CMAQ) Modeling System*, EPA Report, EPA/600/R-99/030, NERL, Research Triangle Park, NC, 1999.

[9] D. W. BYUN, J. PLEIM, R. TANG, AND A. BOURGEOIS, *1999: 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.

[10] D. W. BYUN, K. SCHERE, *EPA's Third Generation Air Quality Modeling System: Description of the Models-3 Community Multiscale Air Quality (CMAQ) Model*. Journal of Mech. Review, 2004

[11] CAO, J., S. A. JARVIS, S. SAINI, AND G. R. NUDD, *GridFlow: Workflow Management for Grid Computing,* In Proceedings of 3rd International Symposium on Cluster Computing and the Grid, at Tokyo, Japan, May 12-15, 2003, p. 198.

[12] B. M. CHAPMAN, B. SUNDARAM, K. THYAGARAJA, S.W. MASOOD, P. NARAYANASAMY, *EZGrid system: A Resource broker for Grids,* http://www.cs.uh.edu/ẽzgrid

[13] B.M. CHAPMAN, Y. LI AND B. SUNDARAM, AND J. HE, *Computational Environment for Air Quality Modeling in Texas,* Use of High Performance Computing in Meteorology, World Scientific Publishing Co, 2003

[14] B.M. CHAPMAN, H. DONEPUDI, J. HE, Y. LI, P. RAGHUNATH, B. SUNDARAM AND Y.YAN, *Grid Environment with Web-Based Portal Access for Air Quality Modeling,* Parallel and Distributed Scientific and Engineering Computing, Practice and Experience, 2003

[15] A. CHERVENAK, I. FOSTER, C. KESSELMAN, C. SALISBURY, S. TUECKE, *The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets,* Journal of Network and Computer Applications, 2001.

[16] K. CZAJKOWSKI, S. FITZGERALD, I. FOSTER, C. KESSELMAN, *Grid Information Services for Distributed Resource Sharing,* Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[17] K. CZAJKOWSKI, I. FOSTER, N. KARONIS, C. KESSELMAN, S. MARTIN, W. SMITH, S. TUECKE, *A Resource Management Architecture for Metacomputing Systems,* Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[18] K. CZAJKOWSKI, I. FOSTER, C. KESSELMAN, *Co-allocation Services for Computational Grids,* Proceedings of the 8th IEEE Symposium on High Performance Distributed Computing, 1999.

[19] W. F. DABBERDT, M. A. CARROLL, D. BAUMGARDNER, G. CARMICHAEL, R. COHEN, T. DYE, J. ELLIS, G. GRELL, S. GRIMMOND, S. HANNA, J. IRWIN, B. LAMB, S. MADRONICH, J. MCQUEEN, J. MEAGHER, T. ODMAN, J. PLEIM, H. P. SCHMID, D. L. WESTPHAL, *Meteorological research needs for improved air quality forecasting,* Report of the 11th Prospectus Development Team of the U.S. Weather Research Program., Bull. Amer. Meteor. Soc., 85, 563-585. 2004.

[20] ERWIN, D. W. AND D. F. SNELLING, *UNICORE: A Grid Computing Environment,* Lecture Notes in Computer Science, Vol. 2150: pp. 825-34, 2001.

[21] I. FOSTER AND C. KESSELMAN, *Globus: A metacomputing infrastructure toolkit,* International Journal of Supercomputer Applications, Summer 1997.

[22] I. FOSTER AND C. KESSELMAN, *The GRID: Blueprint for a new Computing Infrastructure,* Morgan Kauffman Publishers, 1999.

[23] I. FOSTER, C. KESSELMAN, *The Globus Project: A Status Report,* Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop, pp. 4-18, 1998.

[24] I. FOSTER, C. KESSELMAN, G. TSUDIK, S. TUECKE, *A Security Architecture for Computational Grids,* ACM Conference on Computers and Security, 1998, 83-91.

[25] I. FOSTER, C. KESSELMAN, S. TUECKE, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations,* International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.

[26] I. FOSTER, C. KESSELMAN, J. NICK, S. TUECKE, *The Physiology of the Grid: An open grid services architecture for distributed systems integration,* http://www.globus.org/ogsa

[27] G. GRELL, J. DUDHIA, AND D. STAUFFER, *A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5) NCAR/TN-398+STR,* NCAR Tech Notes http://www.mmm.ucar.edu/mm5/

[28] P. GUTMANN, *Plug-and-Play PKI: A PKI Your Mother Can Use,* Proceedings of the 12th USENIX Security Symposium, Washington, DC, August 2003, pp 45-68

[29] P. GUTMANN, *How to build a PKI that works,* Keynote address, 3rd Annual PKI R&D workshop, http://middleware.internet2.edu/pki04/proceedings/pki_that_works.pdf

[30] M. R. HOUYOUX, J. M. VUKOVICH, C. J. COATS, JR., N. W. WHEELER, AND P. S. KASIBHATLA, *Emission inventory development and processing for the seasonal model for regional air quality (SMRAQ) project,* J. Geophys. Res., Atmospheres, 105, D7, 9079-9090. 2000.

[31] S. KRISHNAN, R. BRAMLEY, D. GANNON, M. GOVINDARAJU, J. ALAMEDA, R. ALKIRE, T. DREWS, AND E. WEBB, *The XCAT Science Portal,* in Supercomputing, 2001.

[32] G. VON LASZEWSKI, I. FOSTER, J. GAWOR, W. SMITH, AND S. TUECKE, *CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids,* ACM 2000 Java Grande Conference, 2000.

[33] G. VON LASZEWSKI, IAN FOSTER, JAREK GAWOR, AND PETER LANE, *A Java Commodity Grid Kit,* Concurrency and Computation: Practice and Experience, vol. 13, no. 8-9, pp. 643-662, 2001, http:/www.cogkits.org/

[34] J. LINN, *Generic Security Service Application Program Interface, Version 2, Update 1,* IETF RFC 2743, 2000. http://www.ietf.org/rfc/rfc2743.

[35] LITZKOW, M., LIVNY, M. MUTKA, *Condor - A Hunter of Idle Workstations.* In Proceedings of the 8th International Conference of Distributed Computing Systems, San Jose, USA, 13-17 June 1988

[36] M. LORCH AND D. KAFURA, *Symphony - A Java based Composition and Manipulation Framework for Computational Grids,* in 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, Berlin, Germany, May 2002, pp. 2124.

[37] B. MOHR, F. WOLF, *KOJAK - A Tool Set for Automatic Performance Analysis of Parallel Programs,* Proc. of the European Conference on Parallel Computing, Springer-Verlag, Klagenfurt, Austria, LNCS, 2790, pp. 1301-1304, August 26-29, 2003.

[38] 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

[39] J. NOVOTNY, S. TUECKE, V. WELCH, *Initial Experiences with an Online Certificate Repository for the Grid: MyProxy,* 2001.

[40] A. RAJASEKAR, M. WAN, R. MOORE, W. SCHROEDER, G. KREMENIK, A. JAGATHEESAN, C. COWART, B. ZHU, S. Y. CHEN, R. OLSCHANOWSKY, *Storage Resource Broker: Managing Distributed Data in a Grid,* Computer Society of India Journal, Special Issue of SAN, Vol. 33, No.4, pp. 42-54, Oct 2003.

[41] J. STEINER, B. C. NEUMAN, J. SCHILLER, *Kerberos: An Authentication System for Open Network Systems,* Proceedings of Usenix Conference, 1988, 191-202.

[42] B. SUNDARAM, B. M. CHAPMAN, *Policy Engine: A Framework for Authorization, Accounting Policy Specification and Evaluation in Grids*, 2nd International Workshop on Grid Computing, Nov 2001.

[43] Y. TAHA, A. HELAL, K. AHMED, J. HAMMER, *Managing Multi-Task Systems Using Workflow*, International Journal of Computers and Applications (IJCA), 21:3, pages 69-78, September 1999.

[44] S. TUECKE, D. ENGERT, I. FOSTER, M. THOMPSON, L. PEARLMAN, L. C. KESSELMAN, *Internet X.509 Public Key Infrastructure Proxy Certificate Profile*, IETF Draft draft-ietfpkix-proxy-06.txt, 2003.

[45] J. 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

[46] P. WAGSTROM, S. KRISHNAN, AND G. VON LASZEWSKI, *GSFL: A Workflow Framework for Grid Services*, in SC2002, Baltimore, MD, 11-16 Nov. 2002, (Poster).

[47] R. WILHELMSON, K. DROEGEMEIER, S. GRAVES, M. RAMAMURTHY, D. HAIDVOGEL, B. JEWETT, J. ALAMEDA AND D. GANNON, *Modeling Environment for Atmospheric Discover*,y MEAD Preprint - Annual American Meteorological Society Meeting - February, 2003

[48] R.B. YATES, B.R. NETO, *Modern Information Retrieval*, Addison Wesley Longman Publishing Co. Inc., First Edition, 1999.

[49] Air Quality Modeling Project, University of Houston, http://www.math.uh.edu/aqm

[50] Ant, a Java-based Build Tool, Available: http://ant.apache.org

[51] The Apache Software Foundation, Available: http://www.apache.org

[52] BPEL4WS: Business Process Execution Language for Web Services Version 1.0, http://www.106.ibm.com/developerworks/webservices/library/ws

[53] Campus Grid for Academic Research, Texas Advanced Computing Center, http://www.tacc.utexas.edu

[54] Common Component Architecture, Available: http://www.cca-forum.org

[55] Comprehensive Air quality Model with extensions (CAMx): http://www.camx.com/, 2003

[56] Cookie Signer (CoSign) Single Sign-on solution, http://www.weblogin.org

[57] CrossGrid Meteorological Application, http://www.eucrossgrid.org/Presentations/IST2002_meteo.zip

[58] DAGMAN (DIRECTED ACYCLIC GRAPH MANAGER), 2002, Available from http://www.cs.wisc.edu/condor/dagman.

[59] Distributed Resource Management Application API, http://www.drmaa.org/

[60] Environmental Protection Agency, http://www.epa.gov

[61] eXtensible Markup Language, http://www.w3c.org/XML

[62] File Transfer Protocol, RFC 959, http://www.faqs.org/rfcs/rfc959.html

[63] Global Grid Forum (GGF), http://www.gridforum.org

[64] Grid Computing - COSC6376, University of Houston, http://grid.hpctools.uh.edu/

[65] Grid Physics Network, GriPhyN, http://www.griphyn.org

[66] GSI-enabled OpenSSH, http://grid.ncsa.uiuc.edu/ssh/

[67] High Performance Computing Across Texas (HiPCAT), http://www.hipcat.net

[68] High Performance Computing Center, University of Houston, http://www.hpcc.uh.edu

[69] Hypertext Transfer Protocol, HTTP 1.1, RFC 2616, ftp://ftp.isi.edu/in-notes/rfc2616.txt

[70] Java API for XML Processing, http://java.sun.com/xml/jaxp/index.html

[71] Java Servlet Technology, http://java.sun.com/products/servlet/

[72] Karajan Project, http://karajan.jini.org/

[73] Kerberos Leveraged PKI, KX509, http://www.citi.umich.edu/projects/kerb_pki/

[74] Lightweight Directory Access Protocol, http://www.openldap.org

[75] Nasa Information Power Grid (IPG), http://www.ipg.nasa.gov

[76] National Weather Service: National Air Quality Forecast Capability http://www.nws.noaa.gov/ost/air_quality/.

[77] National Centers for Environmental Prediction (NCEP), http://www.ncep.noaa.gov/

[78] NCEP ETA analysis and forecast http://www.emc.ncep.noaa.gov , http://www.emc.ncep.noaa.gov/data

[79] National Exposure Research Laboratory (NERL), http://www.epa.gov/nerl/

[80] National Oceanic and Atmospheric Administration (NOAA), http://www.noaa.gov/

[81] The Open Grid Computing Environment (OGCE), http://www.ogce.org/references.php

[82] Open Grid Services Infrastructure to WS-Resource Framework: Refactoring and Extension, http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf

[83] Open Grid Services Infrastructure (OGSI) Specification for OGSA services, http://www.gridforum.org/Meetings/ggf7/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf

[84] OpenPBS Batch Processing and Resource Management System, http://www.openpbs.org/

[85] Public Key Infrastructure Standards, http://csrc.nist.gov/pki/panel/warwick

[86] Community Scheduler Framework, http://www.platform.com/products/Globus/

[87] Load Sharing Facility, Resource Management and Job Scheduling System, http://www.platform.com/products/HPC/

[88] Resource Specification Language, RSL, http://www.globus.org/gram/rsl_spec1.html

[89] Secure Sockets Layer Specification 3.0, http://www.netscape.com/eng/ssl3

[90] Simple Object Access Protocol Specification, SOAP Specification version 1.2 http://www.w3.org/TR/soap12/

[91] Sun Grid Engine, Sun Microsystems, http://www.sun.com/software/gridware

[92] Texas Commission on Environmental Quality (TCEQ), http://www.tceq.state.tx.us/index.html

[93] TERAGRID project of NSF, http://www.teragrid.org

[94] Triana Workflow, Web Page. [Online]. Available:http://www.triana.co.uk

[95] Web Services Description Language Specification, WSDL version 1.1, http://www.w3.org/TR/wsdl

[96] Web Services - Resource Framework, Specifications of the WS-Resource construct, http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[97] Weather Research and Forcasting Model (WRF), http://wrf-model.org

[98] X-509 Certificate Format, http://www.w3.org/PICS/DSig/X509_1_0.html