Portlets for User Centric Job and Task Monitoring for Open Science Grid Virtual Organizations

D Alexander, R Pundaleeka, S Tramer
Tech-X Corporation
5621 Arapahoe Avenue Suite A
Boulder, CO 80301
+1 (303) 448-7751
alexanda@txcorp.com, roopa@txcorp.com, tramer@txcorp.com

J Lauret, V Fine
Physics Department
Brookhaven National Laboratory
Upton, NY 11973-5000
+1 (631) 344-2450
jlauret@bnl.gov, fine@bnl.gov

ABSTRACT
Organizations in the Open Science Grid are motivated to provide services so that individual scientists can effectively execute data analysis jobs and take advantage of Grid resources. For success, these scientists will need to be able to monitor the execution status and application-level messages of their jobs that may run at any site within the Virtual Organization. In the Open Science Grid, the existing grid-wide tools provide abundant information about the whole system, but are geared towards Grid administrators, while information tailored towards individual users is not readily available except for the larger organizations that can afford to build their own tools. The User Centric Monitoring project addresses this gap by targeting a novel complete set of user-centric information including the status of a task of submitted jobs, queue positions, times of the start and finish, output and error messages, and reasons for failure. The toolkit is designed to provide flexible collection mechanisms and rich portlet-based presentation tool. This paper will describe the toolkit and portlet designs with a description of the most recent implementation progress.

Categories and Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *web-based services.* C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Distributed applications, Distributed databases.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *User-centered design*

General Terms
Management, Design, Experimentation

Keywords
Grid computing, job monitoring, portlet.

1. INTRODUCTION
The User Centric Monitoring project aims to develop a Monitoring Information Service Toolkit (MIST) that allows a Virtual Organization (VO) within Grids to build secure systems that provide rich sets of intuitive information to their scientists accessible through a Web browser so that these scientists can monitor their computing tasks distributed throughout the Grid and thus, be more productive. MIST is composed of a multi-language software library, an abstracted data store, and a Web portlet repository. The toolkit can be used to augment any Grid system including the Open Science Grid (OSG) and TeraGrid because a multi-language library with broad-level functionality is provided along with web-portal-based code that can be integrated into existing or new portals.

What distinguishes this project from other Grid monitoring projects is that rather than concentrating on monitoring information from an administrative-centric point of view by collecting information that shows stats such as the "up/down" site status, aggregate job information for accounting, or the overall health of the VO's Grid resources, we concentrate on user-centric monitoring information such as how are my jobs doing for a given task, have my jobs reached the queue, what position in the queue are they, which jobs have started, which jobs have stopped, if any jobs have failed, and what are the application level messages associated with my job. It is challenging to collect this type of information from the varying points in the Grid system, but it is crucial information to provide to the users for the ultimate success of the Grid.

There are some excellent administrative monitoring systems in place within some of the Grid systems. For TeraGrid, there is Inca [1]. Inca concentrates on important system administrative aspects such as software stack validation and verification; network bandwidth measurements, and Grid benchmarking. For the OSG, there is GRATIA [2]. GRATIA is an accounting project that tracks jobs and resource usage by VO and site within the OSG. The implied economic purpose of this system is not to tell the user what is happening with the state of their jobs, but to accurately track resource and service usage in the system as a whole.

Also within the OSG, there is the ARDA Dashboard Project [3] for the four Large Hadron Collider (LHC) experiments at the CERN facility. The Dashboard project is probably the closest analogous project to ours in that it focuses on providing user-centric monitoring type information rather than system administrative information. CERN is fortunate to have great resources and experience with large Oracle databases. This works out well for the LHC experiments, but these systems are often difficult to set up for the smaller VOs in the OSG. Also, to date they have not provided application monitoring, although may have designs for that aspect. The MIST concept, as discussed in more detail below, is not an attempt to replace these systems, but rather an effort to augment the toolset available to provide monitoring information that the user cares most about. This is especially crucial for the smaller VOs with few software developers.

2. THE MIST CONCEPTS

There are three main concepts for MIST. First, MIST is deployed and used by a VO by using a single versatile application programming interface from the main library, which we call the Tracking Library. The Tracking Library can be used in many places within the VO systems to collect and filter available task and job monitoring information from various resources such as Grid job schedulers, task meta-schedulers, job wrapping scripts, daemon scripts, and data analysis applications. The second MIST concept is the data store structure with abstractions called collections that hold task, job, and job event details. The data store will have a modular access design that is initially implemented in two ways: first as a database and second as a series of log entries in a file. The final concept is that MIST will use portlets to display the information accessed via a Grid Service. For the portlets, the Tracking Library will also double as an access library for the display portlets, which are designed to present the information to the users as they would like to see it. Figure 1 shows the basics concepts to providing the user monitoring information, which are collection, structure, and presentation.

The MIST tools are designed to give the smaller VO with the Open Science Grid a way to easily develop a monitoring system that integrates well within a web portal. Below we give more details about the MIST concepts and tools.

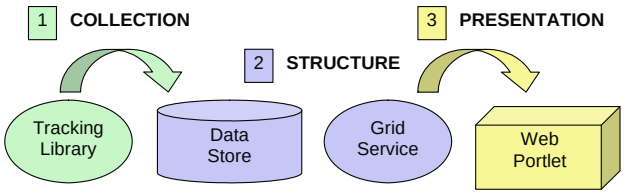

Figure 1. The MIST Toolkit includes tools that help the VO in three steps of user centric monitoring: (1) COLLECTION of the information with a versatile Tracking Library, (2) STRUCTURE of the information with an abstracted Data Store concept for file and database storage, and (3) PRESENTATION of the information with a portlet that can be used within a VO

3. COLLECTION TOOLS IN MIST

The Tracking Library will be called upon in many places in the typical Grid VO use case architecture (see Figure 2) to collect the information need be the user that submits a complex task to a Grid Resource Broker that splits the task into many jobs that may run on various sites within the VO.

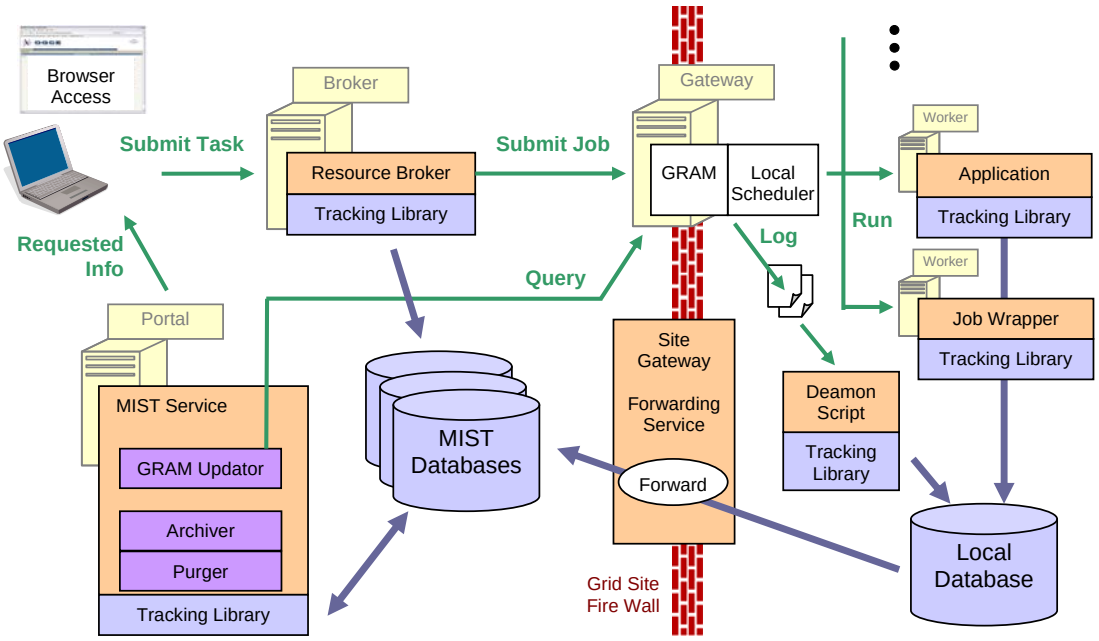

Figure 2. Monitoring information is created in many places in a Grid system. A Tracking Library is proposed that can be used at the resource broker, scheduling, job wrapper, and application levels. The library will feed information in a database (or federation of databases). The monitoring information can also be accessed from the database with the same library.

D Alexander, R Pundaleeka, S Tramer, J Lauret, V Fine, Portlets for User Centric Job and Task Monitoring for Open Science Grid Virtual Organizations, International Workshop on Grid Computing Environments 2007, Nov., Day, 2007, Reno, NV, USA. An electronic version of this document will be available at http://casci.rit.edu/proceedings/gce2007

Because the Tracking Library must be flexible to function in many places in the Grid system, it must have versions in many different languages (Grid middleware solutions which might need to call a Java library, Applications are often will need a C++ version, scripting job wrappers may need a Python version, etc.). To simplify maintenance and development we are maintaining a

single C++ version, upon which we can use SWIG [4] to extend it to other languages such as Java, Python, and Perl.

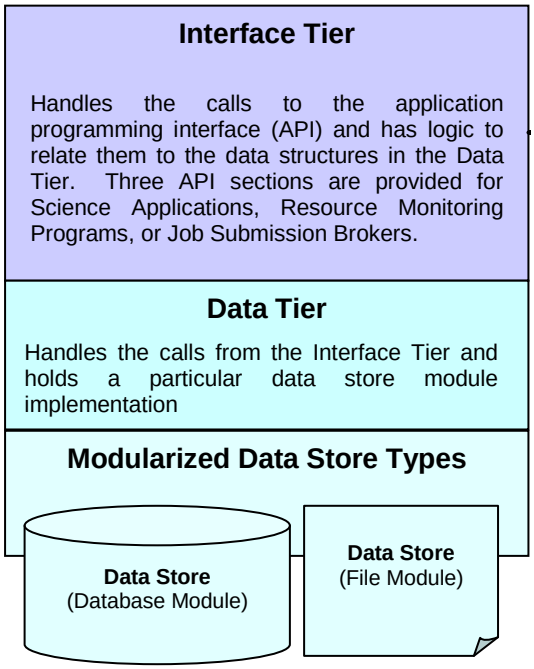

Figure 3. The Tracking Library has a Data Tier that interacts with the MIST data store module interface and an Interface Tier that provides an API for various uses.

API for Science Applications

```
TxEventLog myLog = TxEventLog()

myLog.logSystemEvent(enum stage, enum level,
string systemContext, string systemMsg)

myLog.logUserEvent(enum stage, enum level,
string userContext, string userKey, string
userValue)
```

API for Resource Monitoring

```
TxJob myJob = TxJob(int brokerTaskID, int
brokerJobID)

myJob.setProperty(enum propID, string value)

string val = myJob.getProperty(enum propID)
```

API for Job Submission Brokers

```
TxTask myTask = TxTask(int brokerTaskID, int
brokerID, int requesterID, string name="",
string description="")

myTask.addJob(int brokerJobID)
```

Figure 4. At the bottom level of the library there is an abstracted and modularized notion of a data store with database and file implementations.

The library has a multi-tier design (see Figure 3) and contains an abstract notion of a data store with a modular implementation that can have many representations other than a database. For example, we are engaging the "Center for Enabling Distributed Petascale Science" or CEDPS [5] project, which is charged with the goal of "troubleshooting" jobs within the Grid execution environment and is forming file-based event logging best-practices to which we are aiming to be compatible with. The interface must be broad enough to include use in job submission brokers, programs that monitoring logs of the resource management, and scientific applications. Figure 4 shows part of the interface some of these uses.

4. STRUCTURE TOOLS IN MIST

At the heart of MIST is the data store. An import feature of the overall design is the structure of the data within the store. The MIST data store design is motivated by having the information ultimately be stored in a database, but the design is more flexible and calls for an abstraction of fields, rows, and tables from a database. In our model, an abstracted "storage" object holds many "collections" that hold many records, which in turn hold many "field" values. This is easily mapped to a database, but also can be mapped to other storage methods such as files.

The collections (or tables for database stores) are further defined in MIST by a schema that dictates what information is eventually stored. There are three main collections in MIST: Tasks, Jobs, and Messages. Tasks are sets of jobs as defined by a VO broker for a given scientist making the request. Jobs are connected with the Grid and local scheduling requests as well as application executions. The Messages collection holds the application-level logging and any Grid-wide event that might be collected by the MIST tools. These collections are supplemented by dictionaries collections that translate integer codes for constants such as logging levels, job states, identification descriptions, etc. Figure 5 shows the details of what is contained in the MIST collections.

Individual application execution jobs on the Grid interestingly have more than one identification key to track. There is the key defined by the broker, there is the key returned when submitting to the Grid Resource Location Manager, and finally there is the scheduler queue number. The relationship between these must be tracked throughout the whole system and is one of the fundamental reasons for having the Tracking Library.

The collections have a one-to-one correspondence with database tables if the data store module is a database module. Various parts of the system will fill the collections with information that will be later read out by the presentation portal. The science application will mainly populate Job Events collection, the job submission broker will mainly populate the Tasks collection, and a resource monitoring program that exams the logs of the resource schedulers will mainly populate the Jobs collection.

DATA STORE COLLECTIONS

Tasks
- Task ID (assigned by Broker)
- Broker Task ID (assigned by Broker)
- Broker ID
- Requester ID
- Name
- Description
- Size (number of jobs)
- Remaining size (number of jobs left)
- Submit Time
- Update Time (last time size was updated)

Jobs
- Job ID (assigned by Broker)
- Job ID (assigned by Grid)
- Job ID (assigned by Local Scheduler)
- Task ID (for task that holds this job)
- Grid Submit Time
- Local Scheduler Submit Time
- Site Location
- Queue
- Queue Position
- Node Location
- Start Time
- Update Time (execution state last updated)
- Execution User (local system user)
- State ID (current execution state, 9 defined states)

Job Events
- Job ID (job which generated message)
- Level ID (info, warning, error, etc. 9 defined states)
- Context (bulk category description of event)
- Stage ID (start, status, or end)
- Content (long string of message)
- Event Time

OTHER DICTIONARY COLLECTIONS

Broker, Requester, Stage, Level, State

Figure 5. The Data Store Schema is made up of three main collections: Tasks, Jobs, & Messages.

5. PRESENTATION TOOLS IN MIST

Once the data is in the store, the presentation tools will help the VO build a portal to provide the user with the monitoring information within an environment that is customized to the needs of the VO. In this section we describe the Grid Service and portlet design with in MIST.

We are leveraging efforts like the Open Grid Computing Environment (OGCE) WebSphere-based portal by choosing to display the user-centric monitoring information through JSR-168 compliant portlets [6]. This paper will describe the latest development of portlets that form a portal to submit jobs to a developmental Open Science Grid site and report the monitoring information. Figure 6 shows how individual portlets are combined in a portal. Within MIST, we will be using sample OGCE portlets as templates for developing task submission and user monitoring related portlets.

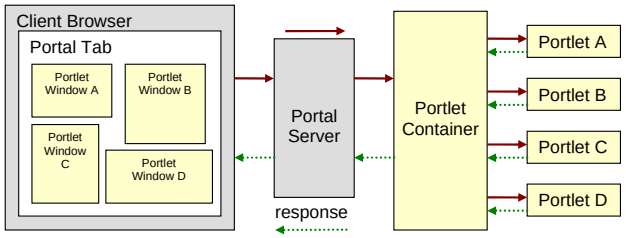

Figure 6. Portlet technology is used for the display of the user information because it can be easily integrated into a portal that has a broader VO function. The MIST portlet can be mixed and matched within the portal with other portlets that, for example, create user proxies and handle user job submission.

The portlet will act as client to the data store when it retrieves the information for the user. To help the VO provide this service, MIST will have a Java client base library to access a Grid Service co-located with the data store. This client base can be reused in a Java application if desired (see Figure 7), but will initially implemented as part of the MIST portlet. This also gives the MIST system a means to securely provide the users with only the information that is connected with their jobs and applications. The data store provides a way to connect all of the identities that are assigned as the work flows through a Grid system (the user's portal ID, the task broker assigned user ID, the Grid certificate, the local execution user, etc.) so that it can all be referenced by the subject of the Grid-issued certificate and the presentation layer can present information on authentication of the user.

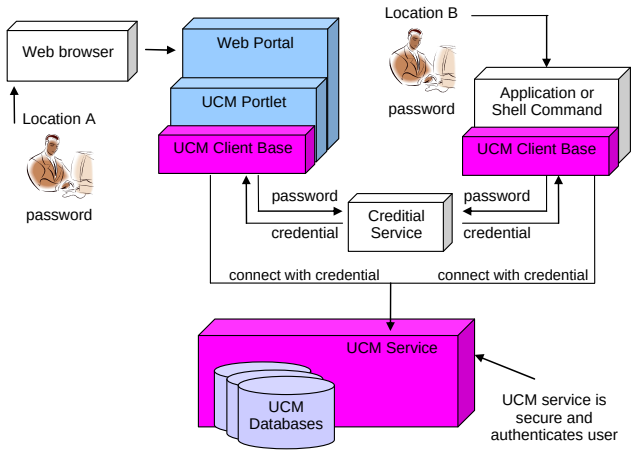

Figure 7. The MIST design is to have a Grid Service and Client Base code that can be embedded in an application or a portlet.

6. Reference Implementation

A reference implementation of the MIST system is underway, being created for the STAR [7] experiment Virtual Organization with the first monitoring site being located at Brookhaven National Laboratory. An implementation of the proposed Tracking Library is complete and being tested. We have completed a working version of the database schema for the data store and have an instance of the database in MySQL [8].

Scalability testing of the Tracking Library and database combination is on-going as the library is being developed.

Initial portlet development is also underway. Work on the MIST portlet has begun using the Open Grid Computing Environment framework that has some existing useful portlets which will allow us to test the portlet in one portal that can obtain a certificate from a MyProxy [9] server and submit a job to a local test Open Science Grid site. Figure 8 shows the first prototype of the presentation portlet and demonstrates the general concepts of our design in that it directly displays the user's information in a task-job-event structure.

We are currently developing portlet code that will garner information from the database that we have in place and display it for users. This code uses Java Server Pages technologies and models after other base portlets that are distributed with the OGCE framework. However, we are also using AJAX [10] technology to separate the calls to the database service and the dynamic interaction the user has with the portlet. This makes the expansion and collapsing of the levels in the display more efficient by removing the constraint of loading a whole portal page for each user mouse click. Full scaling tests remain to be done, but so far the portlet performs well.

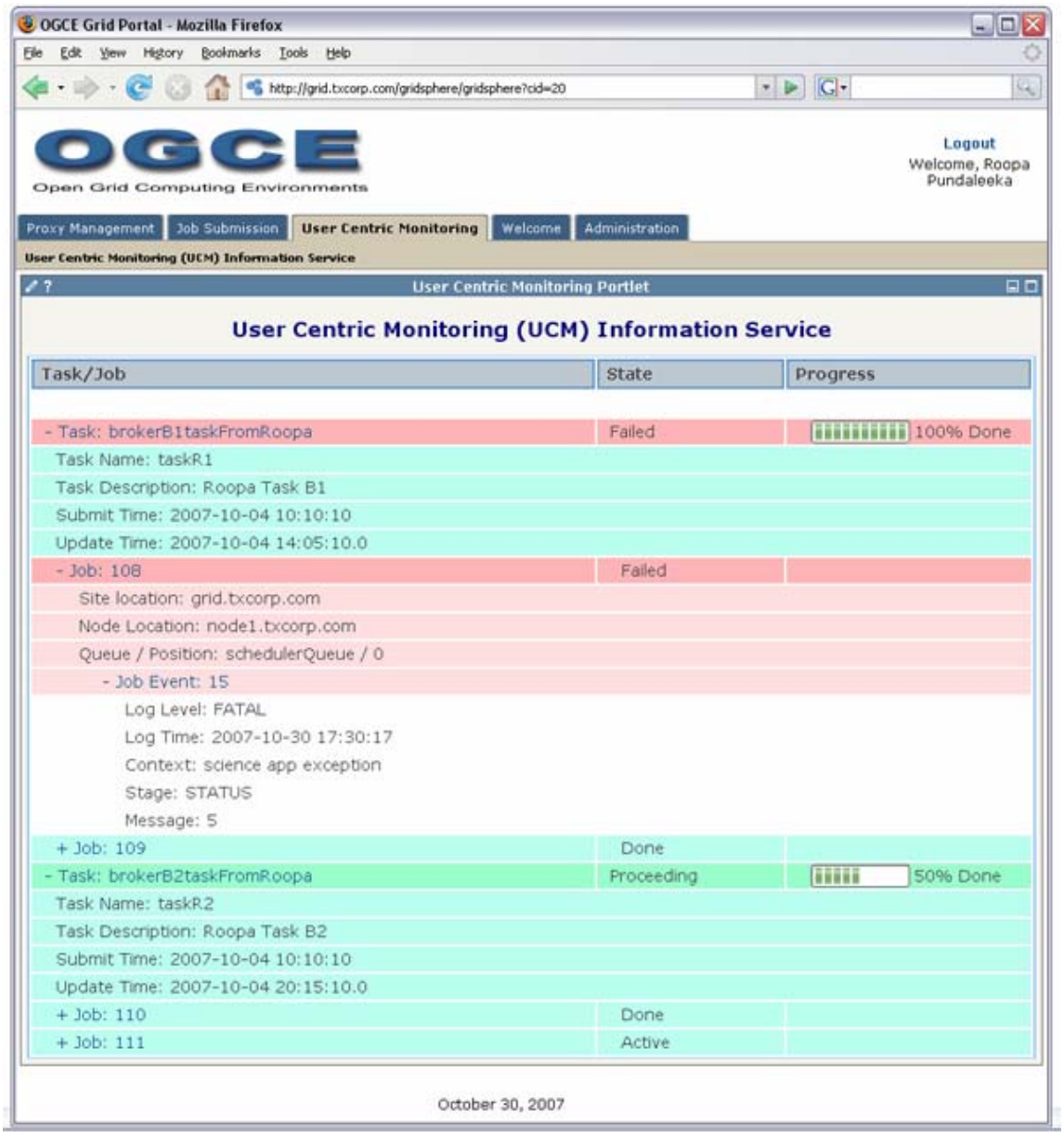

Figure 8. This figure shows the current graphical interface prototype for users to accesses the monitoring information from the Web portal. The user will be able to see their tasks, drill down to individual jobs, and further down to job properties and application-level job events.

7. CONCLUSION

We have described the Monitoring Information Service Toolkit (MIST) that allows the smaller Virtual Organization (VO) within the Open Science Grid to build systems that provide users with a complete picture of what is happening with their complex tasks made up of many jobs. We have shown a portlet design within MIST system that will make it easy to include both job and application monitoring information within a JSR168 compliant portal in a secure fashion.

A reference implementation of the MIST system is underway. The Tracking Library has been written for collecting information at many places in a Grid system such as at the broker that creates tasks composed of many jobs, at the Grid and local resource monitoring systems, and at the application and wrapper-script level. The data store design has remained abstracted to include representations such as database and file, which allows the system to be integrated with efforts such as the event logging systems used at the operating system level. And finally, a prototype portlet has been constructed that displays the data store information. As the project continues, we will build a complete broker-to-application system that shows how MIST can be used within Grids such as the OSG.

8. ACKNOWLEDGMENTS

The work reported here is supported by the U.S. Department of Energy SBIR Grant #DE-FG02-05ER84170.

9. REFERENCES

[1] The Inca Test Harness and Reporting Framework", Shava Smallen, Catherine Olschanowsky, Kate Ericson, Pete Beckman, and Jennifer Schopf, to appear in SuperComputing '04, April 2004. Also available as SDSC Technical Report #SDSC-TR-2004-3. Also, see http://inca.sdsc.edu/

[2] GRATIA, see https://twiki.grid.iu.edu/twiki/bin/view/Accounting/

[3] The ARDA Dashboard, see http://dashboard.cern.ch/

[4] Simplified Wrapper and Interface Generator (SWIG), http://www.swig.org/

[5] The Center for Enabling Distributed Petascale Science (CEDPS) is a US Department of Energy funded SciDAC-2 project to support data placement, scalable services, and troubleshooting in Grid environments. See https://www.cedps.net for more information.

[6] The “JSR 168: Portlet Specification”, http://jcp.org/en/jsr/detail?id=168, defines the interoperability between portlets and portals.

[7] Information about the STAR nuclear physics experiment can be found at http://www.star.bnl.gov/

[8] MySQL is a highly-available, light-weight database server. See http://www.mysql.com/

[9] MyProxy is open source software for managing X.509 Public Key Infrastructure security credentials. See http://grid.ncsa.uiuc.edu/myproxy/

[10] AJAX technology is a web development technique for asynchronously accessing server data while allow the user to interact with an exisiting web pages and updating parts of the page when the data is retrieved. See http://en.wikipedia.org/wiki/Ajax_(programming)