User centric monitoring (UCM) information service for the next generation of Grid-enabled scientists

To cite this article: D A Alexander et al 2008 J. Phys.: Conf. Ser. 119 052001

View the article online for updates and enhancements.

Related content

- PHOTOMETRIC AND MORPHOLOGICAL ANALYSIS OF UCM GALAXIES
  Alvaro G. Vitores Gonzalez

- User Centric Job Monitoring – a redesign and novel approach in the STAR experiment
  D Arkhipkin, J Lauret and Y Zulkameeva

- Using CREAM and CEMonitor for job submission and management in the gLite middleware
  C Aiftimiei, P Andreetto, S Bertocco et al.

Recent citations

- User Centric Job Monitoring – a redesign and novel approach in the STAR experiment
  D Arkhipkin et al

- Automation and quality assurance of the production cycle
  L Hajdu et al
User Centric Monitoring (UCM) information service for the next generation of Grid-enabled scientists

D A Alexander\textsuperscript{1*}, C Li\textsuperscript{1}, J Lauret\textsuperscript{2}, and V Fine\textsuperscript{2}

\textsuperscript{1} Tech-X Corporation, 5621 Arapahoe Avenue Suite A, Boulder, CO 80303, USA
\textsuperscript{2} Physics Department, Brookhaven National Laboratory, Upton, NY 11973, USA
\textsuperscript{*}Email: alexanda@txcorp.com

Abstract. Nuclear and high-energy physicists routinely execute data processing and data analysis jobs on a Grid and need to be able to easily and remotely monitor the execution of these jobs. Existing Grid monitoring tools provide abundant information about the whole system, but are geared towards production jobs and well suited for Grid administrators, while the information tailored towards an individual user is not readily available in a user-friendly and user-centric way. Such User Centric information includes monitoring information such as the status of the submitted job, queue position, time of the start/finish, percentage of being done, error messages, standard output, and reasons for failure. We proposed to develop a framework based on Grid service technology that allows scientists to track and monitor their jobs easily from a user-centric view. The proposed framework aims to be flexible so that it can be applied by any Grid Virtual Organization (VO) with various ways of collecting the user-centric job monitoring information built into the framework. Furthermore, the framework provides a rich and reusable set of methods of presenting the information to the user from within a Web browser and other clients. In this presentation, we will give an architectural overview of the UCM service, show an example implementation in the RHIC/STAR experiment context and discuss limitations and future collaborative work.

1. Introduction

Nuclear and high-energy physicists are increasingly executing individual jobs within a Grid environment under the umbrella of their Virtual Organization (VO). These jobs are not necessarily associated with production data processing or analysis done at large scales, but can be jobs defined by the user for their personal analysis needs. In addition, these jobs may be executed outside the frameworks that are built by a particular experiment with applications that are written by the user. Furthermore, the users may be in a VO that does not have large development efforts for individual frameworks and monitoring systems. The result is that users may be left with no easy way to ascertain job success or failure and separate out failures of the Grid infrastructure from failures within their application. The pioneering VO with large resources can provide users with more of the full spectrum of monitoring information, but the smaller VO is left with very little tools and as a consequence their users have large hurdles to overcome to use Grid resources.

There are many existing Grid monitoring tools from the Grid organizations and middleware providers such as The Open Science Grid [1], GridX1 [2], and TeraGrid [3]; all of which use some form of the Virtual Data [4] and/or Globus [5] Toolkits as middleware. These tools have services that
give abundant information about the systems as a whole, but are geared towards production jobs and well suited for Grid administrators, while the information tailored towards an individual user is not readily available in a user-friendly and user-centric way. There are recent developments in the area of Meta Scheduling that are addressing the needs of the user such as the Gavia Meta Scheduler [6] and the GridWay Metascheduler [7], but we can do much better especially in the area of application logging.

For the most part, though, the typical administrator-centric type of monitoring information includes “up/down” site status, aggregate job information for accounting, and overall health of the Grid resources. These are very import for the VO administrator, but not necessarily targeted for the user, which have somewhat different needs. The user typically has questions such as:

- How their jobs are doing, and if they fail – why?
- Are the submitted jobs are in the queue? What position? What priority?
- When the jobs start? When did they stop? How long do they run?
- What is percentage done of a task consisting of many jobs?
- If job fails, at what level (Grid submission vs. local scheduling vs. application)?
- If at the application, what is the detailed information from the application’s logs?

The ideal system would provide all of this information in a nice human readable format. The User Centric Monitoring (UCM) project [8] aims to develop an information service toolset that provides a rich set of intuitive information to scientists accessible through a Web browser which allows them to monitor their computing tasks distributed throughout the Grid and thus, be more productive. The toolset consists of a Tracking Library, a data storage model, and a Web Portlets for collection, structure, and presentation of information to help the user know how their jobs are progressing and if not then why they have failed. This paper presents the UCM project to address this need and gives a progress report of the initial development. We organize the description and progress information into three sections: COLLECTION, STRUCTURE, & PRESENTATION.

Figure 1. 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.

2. COLLECTION: A Versatile Tracking Library In Many Languages
Any monitoring system needs to first collect the information in order to be able to serve it to the user. In a Grid environment that has a resource broker scheduling tasks composed of many jobs, there are many places in the system where monitoring information is produced. Thus, any library to collect and
retrieve the information must be versatile in that it needs to be available in many languages and have a high level programming interface. Figure 1 shows a diagram of a sample Grid environment where we show our library, called the Tracking Library, being used by many components.

In the UCM system, the Tracking Library is used as a programming interface to collect the user-centric monitoring information in various places throughout the system. It will be implemented in many languages through a main development branch in C++, where the branch will be used to generate further language versions such as Java, Perl, and Python with the SWIG [9] tool.

The Tracking Library has a multi-tier design where the exposed tier will be the application programming interface (API) that is called from various points in the Grid from the resource broker down to the analysis application itself. At the lowest level of the Tracking Library will be a data model and work on this has begun and presented here. The design of the tracking library depends on an abstract data store and action handler that is shown in figure 2. The data store is made up of fields that are combined into records that are in turn combined into collections of records. A handler object for the stores prepares an action (i.e. select, update, insert, or delete) based on a record pattern that is a partially initialized record object. For example, if the goal is to select a record in a collection where a given field in the record has a certain value, then one would prepare a record pattern that is empty except for the given field, which would have the certain value. When the select action is executed, it will search the collection with the proper record types for all records that have the given field with that certain value. In general, code will open a store; create a pattern of a record; prepare an action getting a corresponding handler for that action; and then execute the action.

The Tracking Library design is intimately connected with the structure of the data within the store and this is the topic of the structure section below.

3. STRUCTURE: An Extensible Abstraction From A Database Schema
At the heart of the UCM system is the data store and an import feature of the overall design is the structure of the data within the store. The UCM data store design is motivated by having the information ultimately be able to put into a database, but the design is more flexible and is an abstraction of fields, rows, and tables from a database. Figure 3 shows a model diagram of the classes used for the data structure. In our model, the storage class holds many collections that hold many records, which in turn hold many field values.
Figure 3. The structure of the data store follows an abstracted model of a database shown here by a UML [10] diagram of the classes that hold the data within the Tracking Library. A TxStorage class holds many TxCollection objects which in turn hold many TxRecord objects which finally hold many TxField objects. A TxVariant object is a generic field value holder that has a type.

Figure 4. Diagram of the database schema being used for the UCM prototype. Three main tables: Tasks, Jobs, and Messages form the basic containers of the monitoring information.
For the part of the data tier that actually does the storage of the data, we have a data module hierarchy of classes where in our initial work we planned to implement both a MySQL [11] database and a file modules, although other modules can be easily implemented. The MySQL module is now implemented and currently under consistency and stress testing. The database schema is complete (figure 4) and has three main tables: Tasks, Jobs, and Messages. There are other auxiliary dictionary tables designed to make the storage and access efficient.

4. PRESENTATION: A Web Approach Using The Portlet Standard

The finally leg of the UCM process is the presentation of the information that is collected. The Tracking Library can be used to extract information through the same code mechanism that was used to store the data. The main approach will be to present the information in a Java JSR168 Portlet [12] compliant format, but through underling base code, VOs can build systems for users to access information either through a Web browser or a Web service client application. We have chosen to focus on a monitoring Portlet interface because it will be reusable within any compliant Portal along side Portlets that can authenticate users and do job submission. This gives VOs flexibility in building a Portal system that suits their needs.

<table>
<thead>
<tr>
<th>Task / Job</th>
<th>Status</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>- TASK: Run 12345 Analysis</td>
<td>Done</td>
<td>100% Done</td>
</tr>
<tr>
<td>- TASK: Run 67890 Analysis</td>
<td>Running</td>
<td>90% Done</td>
</tr>
<tr>
<td>+ Job: SUMS0001</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>+ Job: SUMS0002</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>+ TASK: Run 02345 Analysis</td>
<td>Running</td>
<td>70% Done</td>
</tr>
<tr>
<td>+ TASK: Run 56789 Analysis</td>
<td>Done</td>
<td>100% Done</td>
</tr>
<tr>
<td>- TASK: Run 98765 Analysis</td>
<td>Running</td>
<td>30% Done</td>
</tr>
<tr>
<td>+ Job: SUMS00301</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>+ Job: SUMS00302</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>+ Job: SUMS00303</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>+ Job: SUMS00304</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>+ Job: SUMS00305</td>
<td>Pending</td>
<td></td>
</tr>
<tr>
<td>- Job: SUMS00501</td>
<td>Failed</td>
<td></td>
</tr>
</tbody>
</table>

Work on the UCM Portlet has begun using the Open Grid Computing Environment framework that has some existing useful Portlets which will allow us to test the UCM Portlet in one portal. Figure 4 shows a mockup of what we envision as a graphical interface for the user that 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 messages.

5. Conclusions

We present a toolset that is built from the ground up with the design goal to give smaller Virtual Organizations the tools to build a full end-to-end monitoring system that is focused on their user needs. There are many monitoring systems built by the larger organizations, but are not always easy for the smaller VO to adopt because the monitoring system is either tied to a monolithic software framework or requires complex and expensive software components as the Oracle database software.

The UCM toolset consists of a versatile Tracking Library to collect and retrieve information into an abstract data store. Since the Tracking Library is available in many languages and has a generic logging-like interface it can be used in many place through out the Grid environment where the monitoring information may be generated. Furthermore, since the data store is abstracted, we can
have a database and file implementation that provides us design flexibility. Such flexibility can allow us to buffer writing to the database with file writing and also incorporate troubleshooting messages such as those recommended by the CEDPS [13].

Finally, we have begun work on a Web view of the data store that is based on the Portlet standard so that our work can be easily incorporated into any standard JSR168 compliant Portal.

References
[8] Work supported by the US Department of Energy Small Business Innovative Research Grant #DE-FG02-05ER84170
[13] 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.