Monitoring GridX1

by
Chris Usher
University of Victoria, Dept. of Physics
3800 Finnerty Road
Victoria, BC

Physics Co-op Work Term Report

in partial fulfillment
of the requirements of the Physics Co-op Program

Spring 2006

Chris Usher

Department of Physics and Astronomy
University of Victoria

May 15, 2006

Abstract

Particle physics experiments like those at the Large Hadron Collider[1] at CERN require large amounts of computing power to analyze the huge amounts of data produced. Since the analysis is easily broken into small parts that maybe computed separately, grid computing is used to utilize computing resources spread around the world. GridX1[2] is a Canadian grid computing project utilizing resources across the country to perform particle physics computation. An overview of GridX1's monitoring system and new monitoring website will be given in this report. Some information on work with the latest grid software will also be included.

Contents

List of Figures

1 Introduction

Most experimental and observational along with some theoretical science requires large amounts of calculations. Large datasets from experiments and observations produce more accurate results and more complex simulation allows for greater understanding of the theory behind results. Large amounts of calculation require large amounts of computing power to complete them. High energy physics historically and continues to require large amounts of computing resources. In modern particle colliders , such as the Large Hadron Collider (LHC) at CERN, millions of events per second must be examined on the fly to find the hundreds of events per second that are stored for analysis. Storing these event requires hundreds of megabytes per second adding up to petabytes of data a year, equivalent to millions of CDs of data per year. Each event must be reconstructed and further analyzed before being used along with many others to make a measurement.

The amount of computing power and storage space required to analyze and store the data from the LHC exceeds the ability of any one super computer or computing cluster to handle. Since each event may be analyzed independently, analysis and event reconstruct do not require expensive and complicated interprocess communication: they merely require a large number of processors to complete a large number of jobs. With high speed network connections to transfer the input and output, the large numbers of needed processors can be located anywhere. To allow a large number of under utilized computing resources around the world to attack the massive amount of processing need, a grid is being used.

1.1 Grids

A grid is a heterogeneous collection of distributed resources with out any central control. [4] They may be made up several different types of resources, such as computer clusters, mass storage or even instruments such as telescopes, each with different hardware architectures and different software. Often grid resources are spread over a large area, sometimes spanning continents. Grids are usually not owned or controlled by one single organization, commonly being made up of resources donated or provided by several entities. Grids are on demand computing where users are able to utilize resources when needed while allowing those resources to be used for other tasks at other times.

The name comes from the idea that grid computing is analogous to an electrical grid with sources of power (computer or electrical) connected together and to all users. The user is able to use power when ever they need it with out worrying about setting up a new connection with the resource or even which resource to connect to. Furthermore electrical and computing grids are similar in that the source of the power is often separated from the user by large geographic distances. Grids may also be compared with the world wide web in that they both run over the Internet and in that both hide most of the difficulties of accessing heterogeneous computational or information resources from the user.

1.2 Globus Toolkit

Globus Toolkit [3] is the standard open source grid middleware package, providing the software to tie the many resources and users in a grid together. Without middleware a grid is merely a number of computers, unable to be utilized in an effective fashion; middleware is the infrastructure that allows the grid to work coherently. Developed by the Globus Alliance, an international collaboration to build new grid technologies. Globus Toolkit contains software that manages the submission and running of jobs on grid resources, that transfers data between users and resources, that handles security and that monitors the grid. Most production grids use Globus Toolkit 2.4, a stable and mature release, as there middleware while the latest release is Globus Toolkit 4.0 which is based on web services. Web services are applications that use standardized interfaces to communicate with other applications over the Internet usually by using XML (eXtensible Markup Language, a language for describing data)[5] and by using HTTP (Hyper Text Transfer Protocol, the same protocol used for web pages)[5]. Web services are intended to be hardware, operating system and programming language neutral.

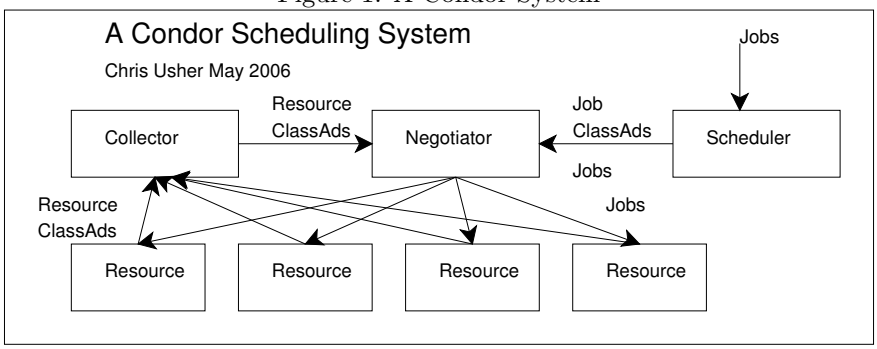
1.3 Condor

Condor[6] is a scheduling system for heterogeneous resources developed by researchers at the University of Wisconsin. Scheduling is the process of matching jobs wanting to run with computing resources best able to run them, then submitting the job to the best computing resource. Jobs usually require certain minimum CPU and memory, operating systems and software packages which must be provided for a given job to run. How quickly a job may be run and how long it may wait is also taken into account in picking were to send a job. In Condor jobs are submitted to the Condor scheduler which produces job ClassAds, descriptions of what the job needs to run. ClassAds are analogous to classified ads in a newspaper, hence the name, advertising what is wanted and what is available. Computing resources publish resource ClassAds detailing the available CPU, memory, operating system and many other resource properties to the Condor collector. The Condor negotiator examines the job and resource ClassAds provided by the scheduler and by the collector and matches jobs with the best place to run them. Next the Condor scheduler submits the job to selected resource. There can be multiple schedulers associated with one collector. Condor-G is an add on to Condor allowing the Condor scheduler to submit to resources through Globus Toolkit. An example of a Condor system may be seen in Figure 1.

1.4 GridX1

GridX1 is a Canadian grid project involving researchers and computing resources at McGill University, National Research Council in Ottawa, the University of Toronto, the University of Alberta, the University of Calgary, Simon

Figure 1: A Condor System



Fraser University, the University of British Columbia, TRIUMF, and the University of Victoria. Currently Linux clusters using either Portable Batch Scheduler (PBS) or Condor scheduling systems are part of GridX1 at the University of Victoria (two clusters), TRIUMF, Simon Fraser University, the University of Alberta, the University of Toronto and McGill University. Most of the clusters, such as mercury2.uvic.ca, are made up of rack mounted blade servers but some such as calliope.phys.uvic.ca are just a bunch of PC networked together, running a local scheduler. GridX1 has access to over two thousand CPUs and several terabytes of storage. In addition to locally scheduled jobs, each GridX1 cluster runs one or both of the two grid applications running on GridX1, BaBar simulations for the BaBar experiment at the Standford Linear Accelerator Center[8], and ATLAS simulations for the ATLAS experiment at LHC[7]. GridX1 uses a pair Condor-G metaschedulers, one for each job type, to schedule and submit, using Globus Toolkit 2.4, jobs to each cluster. The Condor collector, scheduler and negotiator for BaBar is at babargrid.phys.uvic.ca while the collector for ATLAS is at condorg.triumf.ca and the scheduler and negotiator for ATLAS is at atlasgrid.phys.uvic.ca. Currently GridX1 is running over a thousand BaBar jobs a day.

2 Monitoring of GridX1

It is useful for GridX1 administrators and users to view the status of GridX1 resources and the status of computing jobs running on them in one place. In addition an interesting display of the monitoring data can be a useful public relations tool as it shows the grid doing something, a thing that is not always apparent due to the somewhat nebulous nature of the grid.

2.1 monitor.gridx1.ca

All of the monitoring data from GridX1 is available from one web page at monitor.gridx1.ca[9]. Previously the monitoring information was spread over

several pages with in the GridX1 website (www.gridx1.ca) and was presented using the Zope content management system which is currently used to display the rest of the GridX1 website. Due to a wish to combine all of the monitoring data in one place and due the unwieldy nature of Zope, the decision was made to move the monitoring to a single web page using Ajax to dynamically update it. Ajax, an acronym for **A**synchronous **J**avaScript **A**nd **X**ML, is the use of JavaScript[5], the most commonly used web scripting language, to display a web page based on data exchanged with a server asynchronously in XML form. The JavaScript code uses the Document Object Model (DOM)[5] to change how the page is displayed. An Ajax based page can update itself automatically periodically with new information from a server. Cascading Style Sheets (CSS)[5] are used to describe the look of the page, for example the colours of the buttons.

The monitoring page is made up of several sections. First there is the summery with a Google map[10] of the GridX1 resources and a brief description of GridX1. Second there is a outline of how the monitoring page works. Next is the Combined Resource Monitor, a listing of the grid resources and the jobs running on them. Resources outside of Canada that are connected to condorg.triumf.ca may be seen by clicking on the All Resources button. Next is a pair of job lists, showing the details of fifty jobs at a time. For both the resource list and the job lists, each column may be sorted in either ascending or descending order. Next graphs of the number of BaBar jobs run each day and each month are displayed. Lastly, for each GridX1 cluster a section is displayed with information about the status of the cluster.

Apart from the first section of the page which is based on custom code so that the Google map is displayed, the monitoring page is made up of several Elements which are JavaScript objects defined in `core.js`. Each Element may be in three states minimized, semi displayed or maximized. When minimized only a title bar with the name of the section and controls to change the state of the Element is displayed. When semi displayed or maximized the title bar is display along with what ever content dictated by the properties of the Element object. A few Elements also maybe opened in a separate window with same information as when maximized. Each Element has a method, elementDraw which displays the title bar and depending on the state calls one of two methods, one for the semi displayed state and one for the maximized state which uses DOM to display the content of the section.

The monitoring page is made up of two HTML documents,`index.html` and `status.html`; one CSS style sheet, `monitor.css`; four JavaScript files, `core.js`, `map.js`, `schedulers.js` and `site.js` and several images. The webserver is on yamon.phys.uvic.ca which also hosts several of the scripts which generate the monitoring data for the web page while additional monitoring scripts run on gcmon01.phys.uvic.ca. Figures 2, 3 and 4 contains graphical representations of the structure of the monitoring page and the scripts that generate the data.

When the page loads a JavaScript method in `core.js` called *getXML* is called. The method *getXML* makes a XML requests to the resource scripts associated with each scheduler and for cluster information. When all three XML request are complete another method in `core.js`, *newXML*, is called.

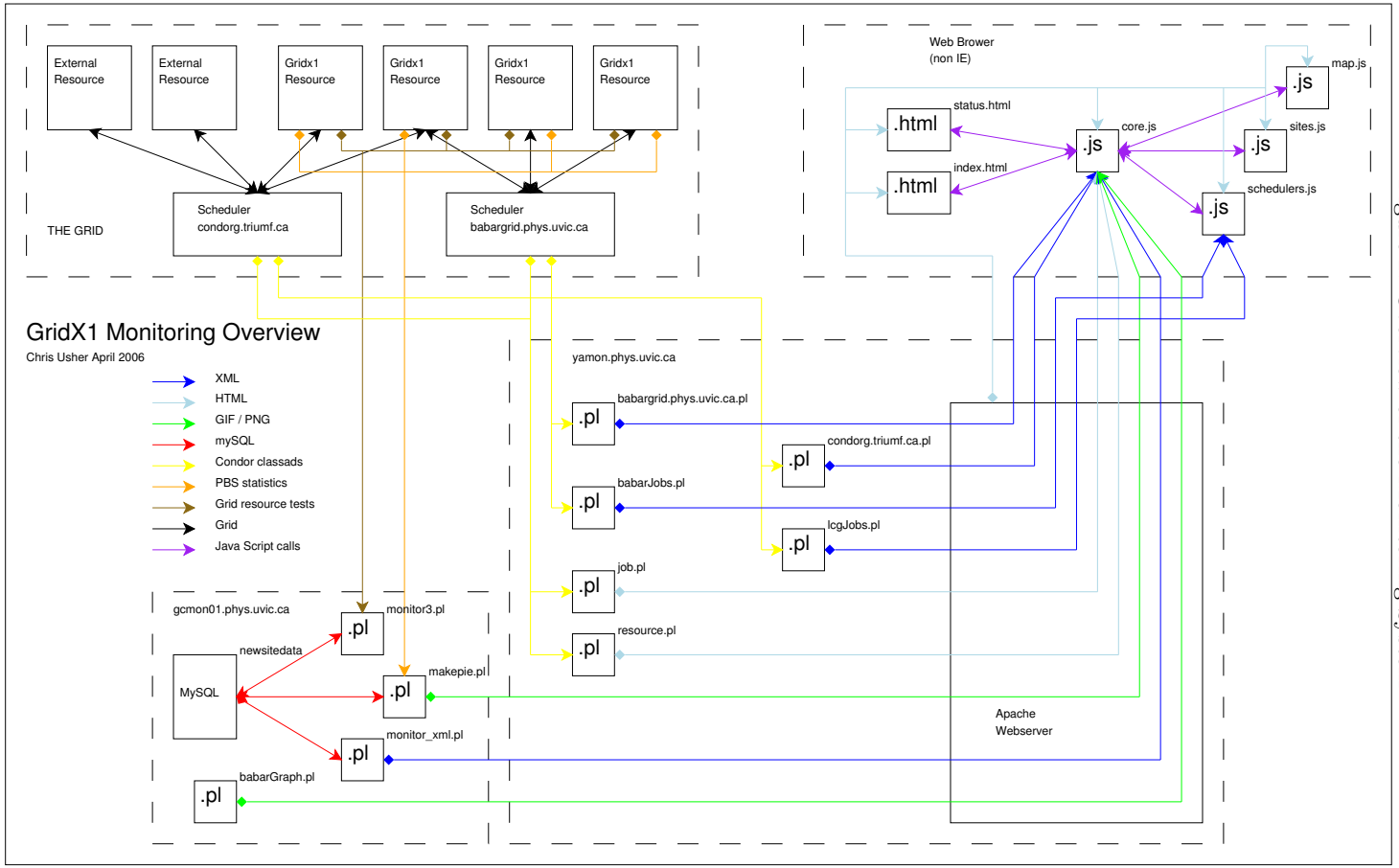


Figure 2: Overview of the Monitoring System

Figure 3: Flow of Control of Monitoring Website. Continued on Next Page

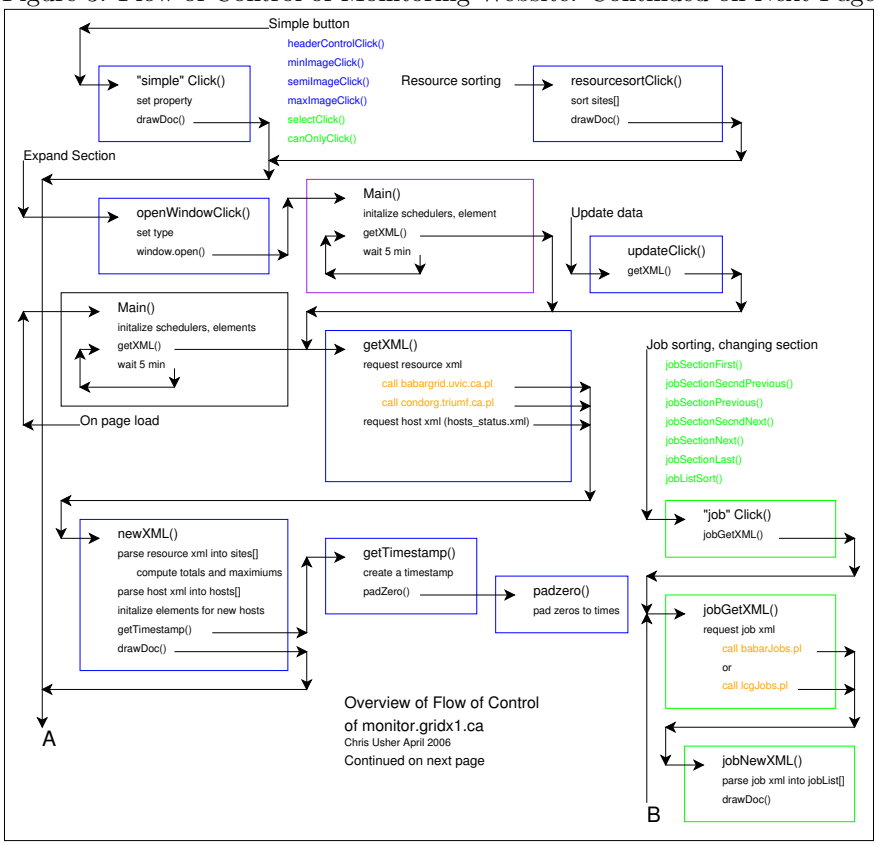
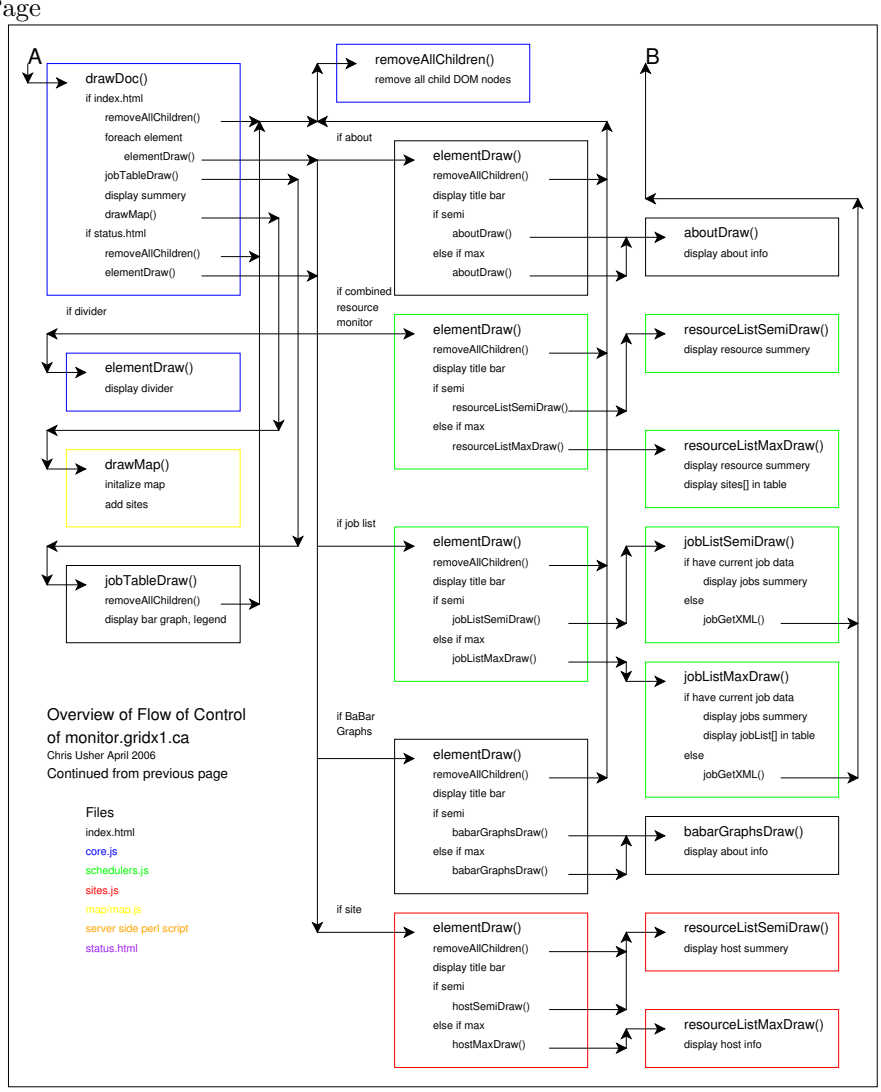


Figure 4: Flow of Control of Monitoring Website. Continued From Pervious Page



This method places the data from the schedulers in one list and the cluster information in another list. Next the `core.js` method *drawDoc* is called. This iterates through each of the Elements and displays each of them as dictated by their state and content. When ever something is changed on the page, usually by a user clicking on a button, *drawDoc* is called. Every five minutes *getXML* is called to update the page with new data while pressing the Update Data button also calls *getXML*.

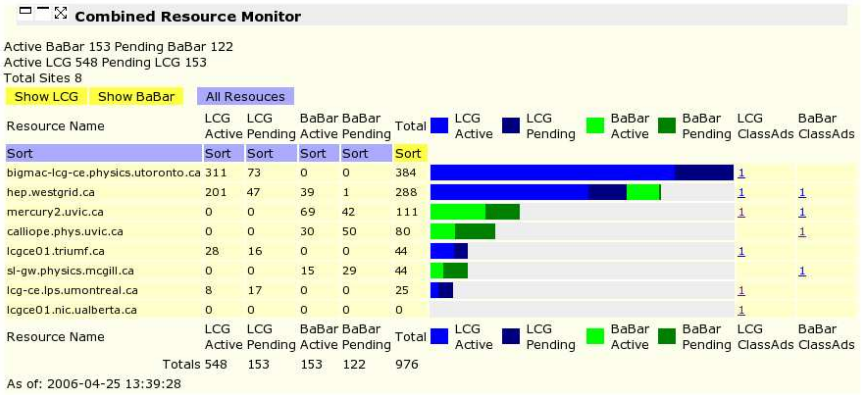
2.2 Combined Resource Monitor

The Combined Resource Monitor provides a listing of grid resources with the number of total, running and pending jobs running on each. A screenshot of the Element may be viewed in Figure 5. For each scheduler a Perl[11] script, `babargrid.phys.uvic.ca.pl` for BaBar and `condorg.triumf .ca.pl` for ATLAS, is run when the monitoring site is loaded to convert the resource ClassAds provided by the Condor command `condor_status -1ong` into XML which is sent to the web browser. The Perl scripts cache the output of the `condor_status` for three minutes so that several calls to the script in a short period of time are not slowed by repeated calls of the relatively slow Condor command. Data from the XML from both of the schedulers scripts is merged into one JavaScript array, both total and maximum jobs are computed, and the array is sorted by an a sorting function in `core.js` by the *newXML* function in `core.js`. Multiply job managers at the same domain are merged into the same line. The *drawDoc* function prints out the array of resource information in table form using DOM with the bar graphs produced with DOM in a function in `core.js`. Several buttons allow the sorting of the table by each of the columns of data displayed, only data from certain schedulers to be shown and for the raw ClassAds to be displayed. When a ClassAd is wanted another Perl script, `resource.pl`, generates the ClassAd generated by calling `condor_status` with the scheduler's and the resource's name. It is returned in HTML which is displayed in a new window.

2.3 Job Monitoring

Each of the schedulers has a job list Element associated with it along with a script, `babarJobs.pl` for BaBar and `lcgJobs.pl` for ATLAS, to generate the XML the Element is based on. A screenshot of an example Element may be viewed in Figure 6. Unlike in the combined resource monitor the XML data is only requested when needed. Since there can be thousands of jobs and only about a hundred resources, running the job scripts when the page initially loads takes way too long. Only if a given job list is semi displayed or maximized will a XML request be made, through the *jobGetXML* method in `schedulers.js`, to the job list script by a method. Data from job ClassAds provided by calling `condor_q -long` is cached for three minutes which in turn is sorted and data on fifty of the jobs are converted to XML. How the data is sorted and which fifty jobs are provided is determined by the arguments supplied to the

Figure 5: Resource Monitor



job script. The returned XML is stored in an array by the method *jobNewXML* in `schedulers.js` which is used by *elementDraw* to create a table with the job details. When a user changes the way the jobs are sorted or which fifty of the jobs are displayed at a given time by clicking on the provided buttons, a new XML request is made and the job script is called with updated sorting and job selection parameters. The data in the job lists are updated every five minutes along with the other monitoring data. The job ClassAd itself may be viewed by clicking on the job ID which calls a Perl script, `job.pl`, which returns the ClassAd in HTML generated from calling `condor_q` with the scheduler's name and job ID which in turn is displayed in a new window.

2.4 BaBar Graphs

Once a day graphs are generated of the number of BaBar jobs that have run each day and each month on the grid. A screenshot of the Element may be viewed in Figure 7. The graphs are created by a Perl script on gcmon01.phys.uvic.ca called `babarGraph.pl` which downloads a text file containing a list of all BaBar jobs that have run. The script counts the number of jobs run at each resource each day and month then uses the GD::Graph Perl package[12] to generate image files of bar graphs of the daily and monthly jobs. The resulting images are then transferred to the monitoring web server, yamon.phys.uvic.ca and are displayed in the monitoring web page.

2.5 Cluster Monitoring

The status of GridX1 clusters is monitored by a few scripts that are run every few minutes on gcmon01.phys.uvic.ca. A screenshot of an example Element may be viewed in Figure 8. Two Perl scripts, `monitor3.pl` and `make_pie.pl`, run a series of tests on each cluster and store the results in a MySQL[13] database, the schema of which may be viewed in Figure 9. Another script, `monitor_xml.pl`

Figure 6: Example of a Job Monitor

BaBar Job List

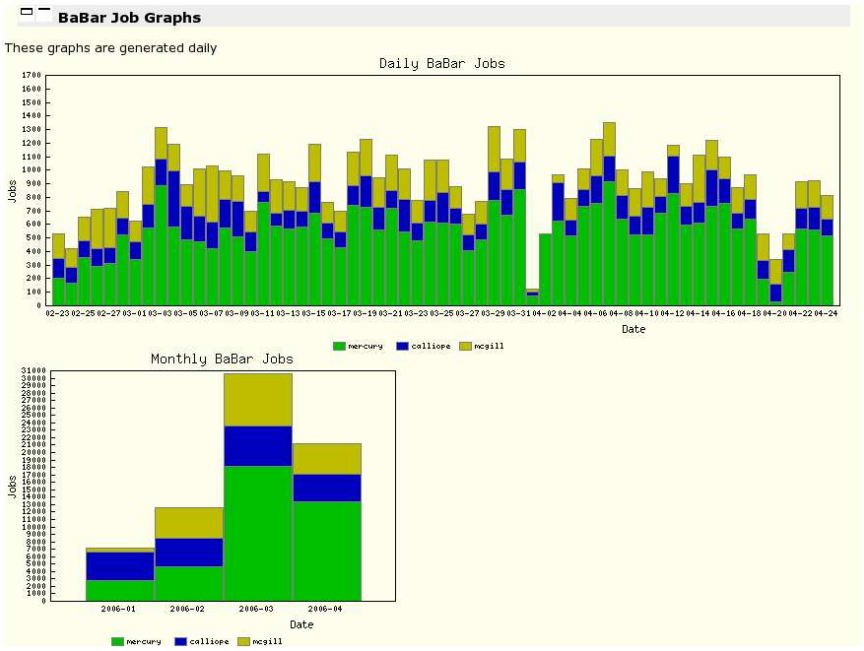
Total Jobs: 235
Active Jobs: 104 Pending Jobs: 117 Done Jobs: 0 Unsubmitted Jobs: 0 Other Jobs: 14
Unmatched Jobs: 0

1 - 50 101 - 150 151 - 200 201 - 235

Job Id	User Id	Owner	Command	Resource	Status	Runtime	Start Time
Sort	Sort	Sort	Sort	Sort	Sort	Sort	Sort
131168.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	PENDING		2006-04-25, 04:31
131167.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	01:46:40	2006-04-25, 04:31
131166.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	01:01:10	2006-04-25, 04:31
131165.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	PENDING		2006-04-25, 04:31
131164.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	00:01:38	2006-04-25, 04:31
131163.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	00:20:10	2006-04-25, 04:31
131162.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	00:00:40	2006-04-25, 04:31
131161.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	01:22:40	2006-04-25, 04:31
131160.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	01:21:40	2006-04-25, 04:31
131159.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	00:05:35	2006-04-25, 04:31
131158.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	02:14:40	2006-04-25, 04:31
131157.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	02:03:36	2006-04-25, 04:31
131156.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:03:40	2006-04-25, 04:31
131155.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	02:37:40	2006-04-25, 04:31
131154.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	02:39:40	2006-04-25, 04:31
131110.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:55:41	2006-04-25, 03:31
131109.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:53:41	2006-04-25, 03:31
131108.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:48:36	2006-04-25, 03:31
131107.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:09:06	2006-04-25, 03:31
131077.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:58:41	2006-04-25, 02:31
131076.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	03:59:41	2006-04-25, 02:31
131075.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:03:36	2006-04-25, 02:31
131074.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:00:41	2006-04-25, 02:31
131045.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:19:06	2006-04-25, 01:31
131044.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:08:36	2006-04-25, 01:31
131043.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:07:41	2006-04-25, 01:31
130987.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:26:09	2006-04-25, 00:31
130986.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:20:07	2006-04-25, 00:31
130985.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:39:20	2006-04-25, 00:31
130984.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:32:17	2006-04-25, 00:31
130980.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:30:12	2006-04-25, 00:31
130979.0	mcgrid	Ashok Agarwal	run.csh 9664 ...'	calliope.phys.uvic.ca	ACTIVE	04:27:10	2006-04-25, 00:31
126489.0	mcgrid	Ashok Agarwal	run.csh 9599 ...'	calliope.phys.uvic.ca	SUSPENDED		2006-04-21, 08:21
126484.0	mcgrid	Ashok Agarwal	run.csh 9599 ...'	calliope.phys.uvic.ca	SUSPENDED		2006-04-21, 08:21
126483.0	mcgrid	Ashok Agarwal	run.csh 9599 ...'	calliope.phys.uvic.ca	SUSPENDED		2006-04-21, 08:21
Job Id	User Id	Owner	Command	Resource	Status	Runtime	Time Submitted

As of: 2006-04-25 13:43:47

Figure 7: BaBar Job Graphs



retrieves the data from database, `newgridinfo`, and converts it to XML which is transferred to the web server yamon.phys.uvic.ca. `monitor3.pl` runs three tests on each Gridx1 cluster. First a test of the Globus gatekeeper is made by trying `globusrun -a -r [cluster URL]/jobmanager-fork`. Next job submission is tested by trying `globusrun -a -r [cluster URL]/jobmanager-fork&(executable=/bin/echo)(arguments=jobtest)`. Lastly gridFTP is tested by running `globus-url-copy file:///tmp/gridftp.test gsiftp://[cluster URL]/tmp/gridftp.test` and `globus-url-copy gsiftp://[cluster URL]/tmp/gridftp.test file:///tmp/gridftp.test2` then checking that the original is the same as the transferred file. If the test does not complete in a given time it is considered failed and the script moves on to the next test. Each test result is stored separately in the database. The script `make_pie.pl` obtains statistics about the number of jobs and CPUs at each cluster and then uses them to create a pie chart of the job load for each. First the script uses Globus to run the PBS command `q_stat` at each of the clusters (at the University of Toronto `condor_q` since they use Condor rather than PBS as their local scheduler) to find the number of grid and local jobs running. Next Globus is used to run the PBS command `pbsnodes` (`condor_status` at the University of Toronto) to generate CPU statistics. Both commands are only given a set time to run before they fail and both commands store their results in the database. From the job and CPU statistics the script generates pie graphs for each cluster showing the number of grid jobs, local jobs and idle CPUs. These are transferred over to

Figure 8: Example of a Clustor Monitor

the web server on yamon.phys.uvic.ca and are displayed on the webpage. The XML data from the script `monitor_xml.pl` is used to create an Element at the bottom of the monitoring page for each cluster with the results of the tests run on that cluster.

2.6 Google Map

Data from both the Combined Resource Monitor and from each cluster are combined in the Google Map at the top of the page. A screenshot of the Element may be viewed in Figure 10. Using their latitude and longitude provided with their status, each cluster is plotted on the map of Canada using the Google Map's API. Users may zoom in and see which buildings the cluster are located in while clicking on a cluster's icon brings up an info window with a summery of the cluster's status.

3 Globus Toolkit 4

Canarie[14], a not-for-profit corporation set up to advance Internet development in Canada, has contracted grid researchers at the University of Victoria and the National Research Council to investigate grid registry and metascheduling services[15]. A grid registry stores information on what resources are available and what jobs are running for the metascheduler to use. In the Condor-G based GridX1 Condor ClassAds are used at the registry. The project involves evaluating registry and metascheduling services used on other grids around the would then developing and deploying an improved registry and metascheduling service on GridX1. Part of the project is developing the services as web services and developing for Globus Toolkit 4, the latest version of Globus Toolkit, as well as for Globus Toolkit 2 which is in current use on GridX1.

3.1 Globus Toolkit 4 Testbed

To gain familiarity with Globus Toolkit 4 and to provide a place to develop and to test the registry and metascheduling services before deploying them on GridX1, a testbed was created. After some early attempts at set up Globus

Figure 9: Monitoring Database Schema

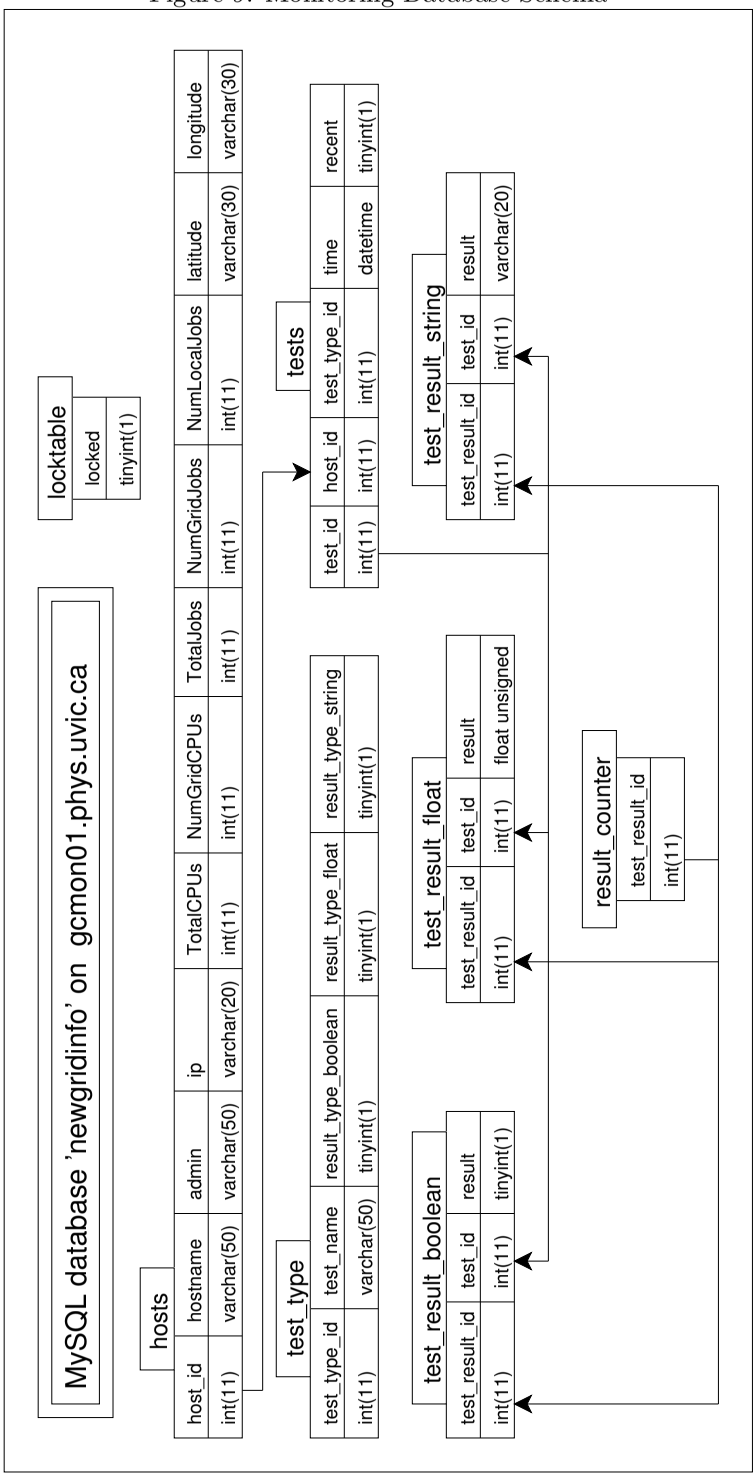


Figure 10: Top of Monitoring Page and Google Map

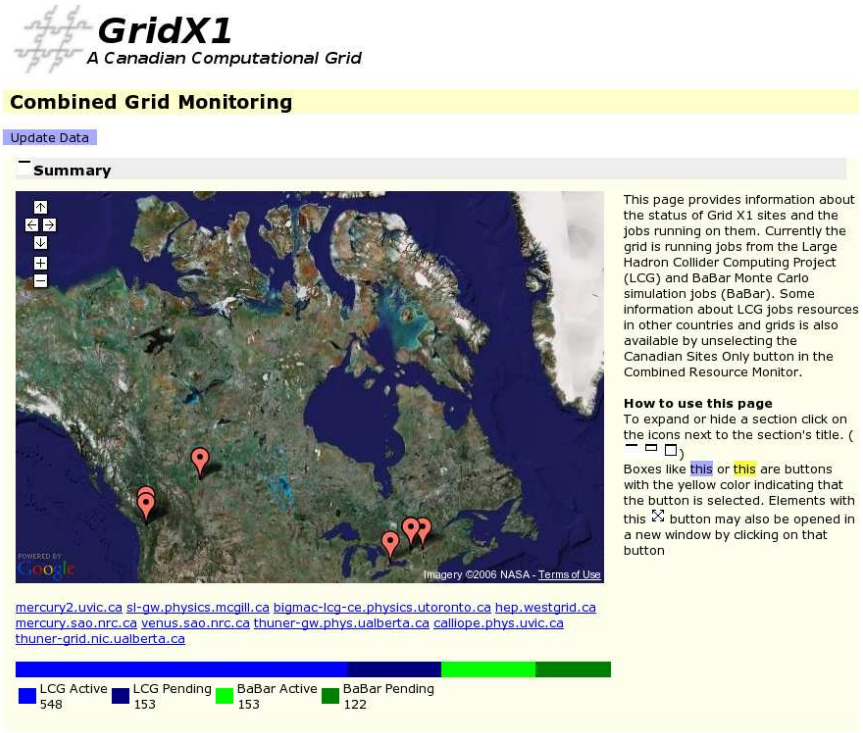
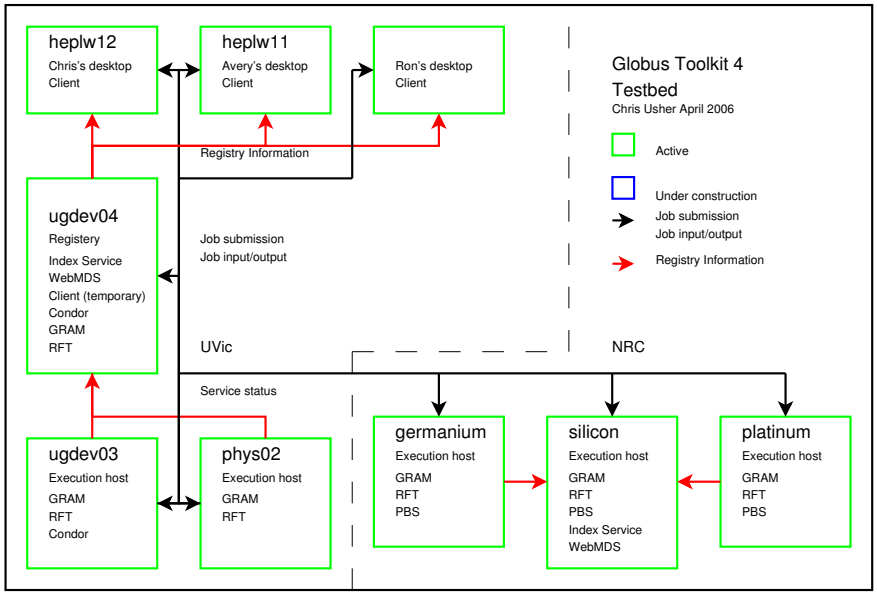


Figure 11: Globus Toolkit 4 Testbed



Toolkit 4, two machines at the University of Victoria, ugdev03.phys.uvic.ca and ugdev04.phys.uvic.ca, were set as a basic testbed. Installing Globus Toolkit 4 on them was a lengthy process as Globus Toolkit 4 is a large, complex collection of software, requireing a large amount of set up and configuration as well as the set up of several software prerequisites. Later Globus Toolkit 4 was also set up on another machine at the University of Victoria, phys02.comp.uvic.ca and Globus clients were installed on several desktops. The job submission, file transfer, security and monitoring features of Globus Toolkit 4 were all tested. Recently the testbed at the University of Victoria was combined with several machines at the National Research Council to form a larger test bed. However there were a few problems having machines at the two sites work together due to the large firewall at NRC. A simple registry was set up on ugdev04.phys.uvic.ca to which the other computers in the test bed report their status. A diagram of the testbed may be viewed in Figure 11.

3.2 Globus Toolkit 4 Test Client

To further test the testbed and to learn something about developing software to work with Globus Toolkit 4 a simple client for it was written. The client, written in Java[16], connects to the registry, a web service called `DefaultIndexService`, and searches through its entries for machines available to run jobs. One of the machines is picked at random and the client loads a XML job description,

detailing what the job is and how it is run. Next the client sets up the security for the job and establishes a connection with the `ManagedJobFactoryService` on the selected machine, the web service that runs the job. Lastly the client actually submits the job to the `ManagedJobFactoryService`. The client does not report whether the job completed successfully since the client does not keep connected to the `ManagedJobFactoryService`. While the client was able to successfully submit jobs to Globus Toolkit 4 machines at the University of Victoria, a few problems were encountered submitting jobs to the machines at NRC due to firewall issues.

4 Conclusion

GridX1 currently runs a large number of particle physics jobs on computer clusters across Canada using Globus Toolkit and Condor. Monitoring data on job and resource status is gathered from GridX1's Condor-G scheduling systems and displayed, along with status the of GridX1 clusters, on the monitoring webpage at monitor.gridx1.ca. Work is already being carried out on the latest grid middleware, Globus Toolkit 4, on GridX1.

5 Recommendations

Currently the monitoring page can only deal with a condor collector having only one scheduler but this is not always the case as a collect can have multiply schedulers. The scheduler JavaScript objects should be broken into to classes, one for schedulers and one for collectors. The combined resource monitor would include data from all the collector objects while for each scheduler object a job table should be displayed. Also it may be an idea to replace the scripts that generate the resource and job XML with a single script for the resources and a single script for the jobs which are passed the Condor scheduler and / or collector as arguments like the two scripts which display the ClassAds. As well the data for the resources and jobs is only as good as the ClassAds for them. If the ClassAds are incorrect then the monitoring will also be.

6 Acknowledgments

I would like to especially thank Dr Randall Sobie for giving me an opportunity to work on Grid Computing at the University of Victoria. I found the work term to be a rewarding learning experience and would recommend it to fellow Coop students. I would also like to thank Dr Ashok Agarwal, Dan Vanderster and Ian Gable as well as Avery Berman, Greg King, Ron Desmarais and Howard Peng.

References

[1] Large Hadron Collider (2006)
http://lhc.web.cern.ch/lhc/

[2] GridX1 (2006)
http://www.gridx1.ca/

[3] Globus Toolkit (2006)
http://www.globus.org/toolkit

[4] I. Foster, What is the Grid? A Three Point Checklist. GRIDToday, July 20, 2002.

[5] For the standards see http://www.w3.org (2006)
For introductions see http://developer.mozilla.org/ (2006)

[6] Condor (2006)
http://www.cs.wisc.edu/condor/

[7] ATLAS Experiment (2006)
http://atlas.web.cern.ch/Atlas/index.html

[8] BaBar Experiment (2006)
http://www.slac.stanford.edu/BFROOT/

[9] GridX1 Monitoring (2006)
http://monitoring.gridx1.ca

[10] Google Maps (2006)
http://www.google.com/apis/maps/

[11] Perl (2006)
http://www.perl.org
See also Larry Wall, Tom Christiansen & Jon Orwant, Programming Perl. O'Reilly, 2000.

[12] GD::Graph Perl package (2006)
http://search.cpan.org/ bwarfield/GDGraph-1.4307/Graph.pm
http://www.boutell.com/gd/

[13] MySQL Database (2006)
http://www.mysql.com/

[14] Canarie (2006)
http://www.canarie.ca/

[15] Computational Grid Services Project (2006)
http://sp.gridx1.ca

[16] Java (2006)
http://java.sun.com/