Running BaBar Applications with a Web-Services Based Grid

by

Tristan Sullivan
University of Victoria
3800 Finnerty Road
Victoria, BC

Physics & Astronomy Co-op Work Term Report

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

Spring 2007

Tristan Sullivan

Department of Physics and Astronomy
University of Victoria

Abstract

In order to help analyze the data produced by the BaBar experiment[1] being conducted at the Stanford Linear Accelerator Centre, the University of Victoria's grid computing group had set up a computational grid using the Globus Toolkit [2]. Since then, a significantly revised version of the Globus Toolkit has been released, and it was desired that the UVic clusters be upgraded to use this newer version. This report includes an overview of the grid, a description of the Globus Toolkit and some of the other software used, some of the details of the upgrading process, and a description of the work required to actually use the upgraded cluster to run BaBar applications.

Contents

List of Figures

1. Introduction:

1.1 The Grid

High-energy physics experiments, and many scientific experiments in general, generate far too much data to be expeditiously analyzed by hand, by a single computer, or even by a single cluster of computers. So, to solve this problem, a number of clusters can be unified into a single computational grid. An obvious issue with this setup is that of scheduling; when a user submits a job to a single cluster, the head node of that cluster decides, based on the job's requirements, to which worker node that job should go. In order to make the grid as analogous as possible, an operation called metascheduling is performed. One machine is designated as the metascheduler, and that is the machine to which users actually submit jobs, and from there the metascheduler sends the job to one of the clusters that make up the grid, again based on the job's requirements. The head node of that cluster then, as per usual, sends the job to one of its worker nodes, on which the job executes. Once the job has been completed, the output is streamed back to the user.

The primary advantage of this approach, as opposed to simply making one very large cluster, is that, generally, all the machines in any given cluster must be homogeneous in terms of operating system and architecture. They also are typically geographically proximate. While these restrictions still apply to any given cluster in the grid, the grid itself can be both far-flung and heterogeneous, since it is nothing but a group of clusters.

There are two obvious ways for the metascheduler to keep track of which clusters it can send jobs to: either the head nodes of the clusters can directly send information to the metascheduler, or they can send the information to a central registry and the metascheduler can then query the registry. It is this latter approach that is in use at UVic. The registry contains a list of what machines have what services available, and a background process running on the metascheduler populates the list of clusters to which jobs can be sent by querying the registry and determining how many machines have the job submission service available.

1.2 The Globus Toolkit[2]

The Globus Toolkit is the software used in GridX1[3], the Canadian grid of which the UVic cluster is a part, to provide the behaviour described above. It provides simple command-line tools for, among many other things, job submission and file transfer. It also provides support for metascheduling, which was described above, though it does not require it; jobs can be directly submitted to any of the individual clusters that make up the grid, or they can be submitted to the metascheduler machine, which will then handle the sending of the job to one of the clusters. The latest version of Globus accomplishes this through the standard web services protocol[4]; job submission, file transfer, and all other features of Globus are services and can be accessed from any machine through the Globus container. The Globus container is a background process that runs on each machine on the grid; all requests made to Globus go through this container. The approach of using a standard protocol is intended to allow for easy extensibility; prior versions of Globus had used a custom protocol[5], but this was found to be insufficiently robust and extensible.

The only one of Globus' many services that was frequently used was Grid Resource Allocation and Management (GRAM), the job submission service. This works by sending the job to a Local Resource Management System (LRMS) specified by the user installed on a cluster specified by the user. Two examples of Local Resource Management Systems are Condor and PBS. Condor, however, can also be used as a metascheduler, as is described in the section about Condor. In either case, the job is submitted to Globus in the same way: with the specification that it should be handled by Condor. What actually happens is determined by whether or not the machine to which the job was submitted was a metascheduler or the head node of a cluster that uses Condor for its LRMS.

1.3 Condor

Condor, a piece of parallel computing software developed at the University of Wisconsin-Madison[6], was originally intended to run on a single cluster. The head node, or Central Manager in Condor parlance[7], would have a list of worker nodes to which it

could send jobs; the Central Manager and worker nodes together form what is known as a Condor pool.

In the case of multiple clusters, Condor can be used as a metascheduler; rather than the Condor pool being populated by worker nodes, it is instead populated by the head nodes of the various clusters that make up the grid. When a user submits a job to the Condor metascheduler, it sends the job to one of the head nodes in the same manner as the Central Manager would normally send the job to one of its worker nodes. From there, the job is sent to one of the worker nodes, and the output from the job is ultimately staged back to the metascheduler and then to the machine from which the user submitted the job.

1.4 The Portable Batch System[8]

The Portable Batch System (PBS) is a Local Resource Management System, meaning that it is intended to be installed on a single computing cluster which consists of many worker nodes and one head node. Jobs are submitted to the head node, which will then send them to the worker nodes. PBS has many features, such as the presence of multiple queues with different behaviours, but there is no need to go into any of them in detail.

1.5 The BaBar Experiment

The BaBar experiment is a high-energy physics experiment being conducted at the Stanford Linear Accelerator Centre (SLAC). To analyze the data it generates requires a large amount of computing power, and as such gridX1 is participating in the analysis. This simply entails submitting a job to the grid that executes a script that runs the analysis, after properly setting up the environment.

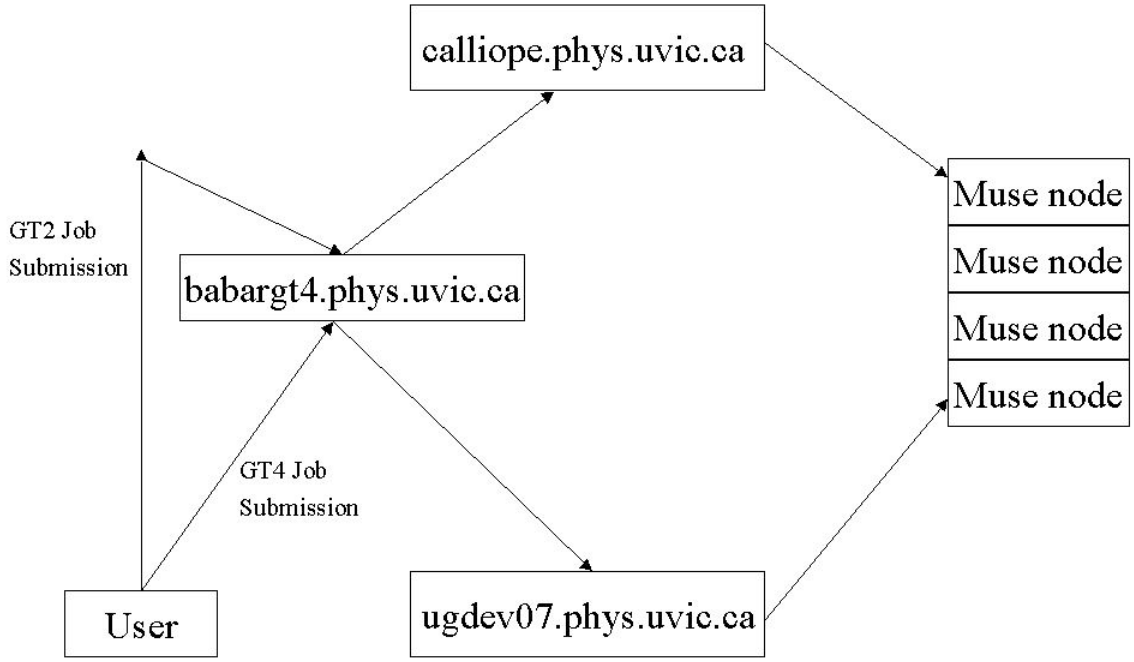
2. Updating From GT2 to GT4

As was mentioned above, old versions of the Globus Toolkit did not use the standard Web-Services protocol, but rather each grid service used its own custom

protocol. This was difficult both to extend and to maintain; difficult to extend because the creation of a new service required the creation of a new protocol, and difficult to maintain because slight changes in the various protocols were often made between versions of the services, and so the software was not properly backwards compatible. By contrast, the latest version of the Globus Toolkit, Globus4, uses the Web-Services Protocol for all the various grid services it contains, such as job submission and file transfer. In order to maintain backwards compatibility, it also supports the use of the older services. In addition to this advantage, it also provides better security, as under the old versions of Globus the job service ran as a privileged user, whereas the job service in the new version of Globus runs as a non-privileged user.

In light of these advantages, the grid computing group at UVic decided to test the functionality of Globus4 on a production-scale grid. The Muse cluster and the Mercury cluster were the two clusters intended to be used for this purpose; however, only the Muse cluster has as yet been set up to run jobs using GT4.

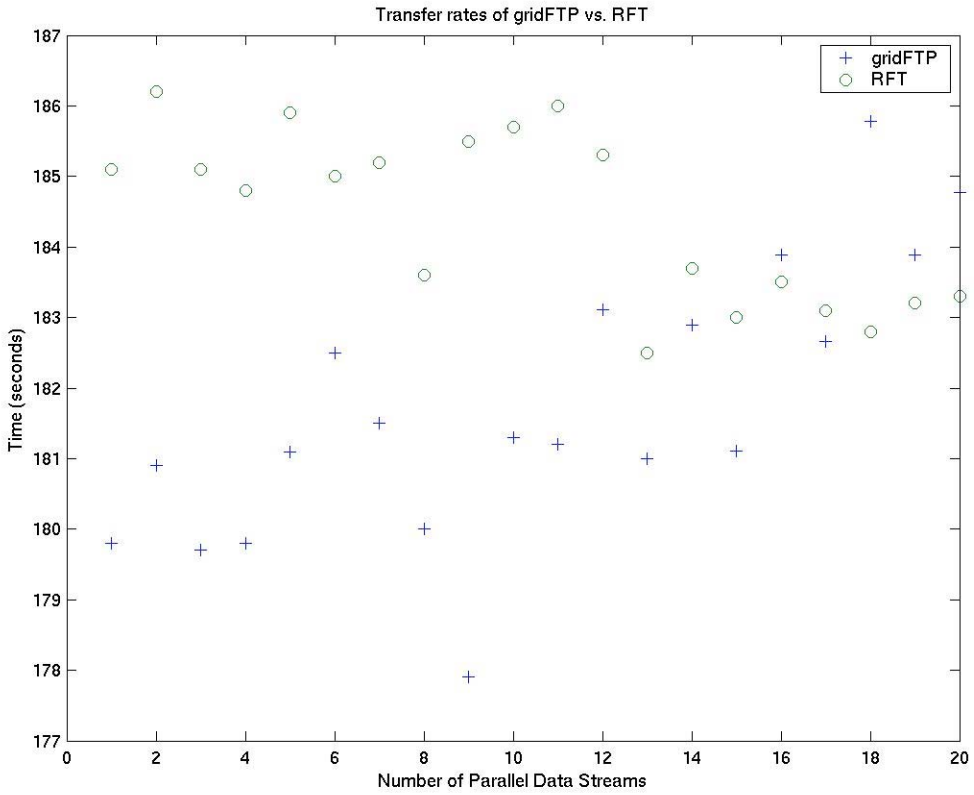
During the setting up of the Muse cluster in order that jobs could be submitted to it using GT4, several complications arose. The most severe of these was that it was required that the current setup not be affected by the upgrade; that is, it was required that jobs could still be submitted to the cluster using GT2. Furthermore, it was required that the head node of the Muse cluster, calliope.phys.uvic.ca, not be altered in any way, as it was pivotal that it remain working properly. It was therefore necessary to set up another machine as an alternate head node to the Muse cluster. The machine that was ultimately selected for this purpose was ugdev07.phys.uvic.ca. The metascheduler for this grid is babargt4.phys.uvic.ca.



The above diagram shows the desired setup for the test grid; once the job has reached either of the head nodes, the transmission from the head node to the worker node is handled identically, as that is done by the Local Resource Management System, which in this case is PBS. From the user's perspective, the only difference is the command used to submit the job. He can either submit it to the metascheduler, in which case he can request that it be submitted to either of the resources, or he can submit it directly to one of the head nodes. In either case, the command will be different, but once the job has reached the head node, the process appears identical. In reality, there are some differences, such as that the transferring of input and output files for the job is handled by a different service, but this should make no difference to the user.

Successfully achieving this setup proved to be surprisingly difficult. In fact, it was not achieved as it was originally intended; instead, it was set up so that ugdev07.phys.uvic.ca simply redirected jobs that were submitted to it to calliope.phys.uvic.ca, and from there they went to the worker nodes. Again, however, from the user's point of view, the functionality is identical.

The upgrade from GT2 to GT4, once it was completed, did not seem to have any effect on the stability of the cluster: jobs that were submitted correctly still typically executed correctly. Occasionally the Globus container on either the metascheduler or the head node would have to be restarted to fix inexplicable job failures, but this is not a time-consuming operation. It is ultimately to be hoped that future releases of Globus will be more stable than the current release. Another concern was that the new Web-Services-based grid services would be slower than their older equivalents. No noticeable performance difference was detected with test jobs; they were simple enough that they executed too quickly in either case to get an accurate sense of the performance difference, if one existed. This was later tested thoroughly for both job submission and file transfer; the details of the job submission tests will be addressed in the section on the submission of actual BaBar jobs. The file transfer test was simple; using both the pre Web-Services file-transfer mechanism (gridFTP) and the Web-Services file-transfer mechanism (RFT), the same file was transferred and the operation was timed. As a separate point of interest, the effect of the number of parallel data streams used in the transfer was measured. From one to twenty data streams were used, with each number of data streams being tested ten times. The times for each were then averaged and the following plot produced.



The size of the file used for the transfer was approximately 56 megabytes. The time difference in transferring a file of this size is quite small, as can be seen from the scale on the y-axis of the graph: less than ten seconds even where the disparity is most extreme. The general trend does seem to be that RFT is slightly slower; however, given that the applications that the grid will be running take several hours at least to finish, a difference of ten seconds in file transfer is negligible.

3. Running BaBar Applications

Once the grid was set up as described above, and simple test jobs were submitted to it, it was used to run test BaBar applications. That is, jobs were run that used the BaBar software, but real input data was not used. The input data is categorised based on run number: different run numbers use different data. The run numbers used for testing purposes were all in the validation range, meaning that the jobs were not processing any

real data. However, since everything else about the submission is identical, if validation runs were made to work then real runs should work also.

A perl script was already present that would, when given a set of run numbers, submit all the runs to the GT2 metascheduler, ugpro03.phys.uvic.ca, and from there they would follow the progression in the above diagram: from the metascheduler, they would go directly to calliope.phys.uvic.ca and then to a worker node. It is a different metascheduler than is indicated in the diagram, but it still submits jobs to calliope.phys.uvic.ca. It was necessary that a script like this be written that would instead submit the jobs to the GT4 metascheduler, babargt4.phys.uvic.ca.

The process is similar conceptually, but the details are actually somewhat different. The original perl script worked by directly running the command to submit a job to Condor, rather than to Globus; this at first seemed rather confusing, but it ultimately produces the same effect as submitting a job to Globus and specifying that it should be handled by Condor, which is the alternative way of submitting a job for metascheduling. However, it was desired that either approach should be possible, so two perl scripts had to be written: one to submit jobs directly to Condor on the metascheduler, and one to submit jobs to Globus, specifying that the job should be handled by Condor on the metascheduler. The syntax for job description is quite different depending on whether or not the job is to be submitted to Globus or to Condor; if it is to be submitted to Condor, the information about the job is written in a syntax specific to Condor, whereas if it is to be submitted to Globus, the information about the job is written in XML[4]. Thus, the two scripts were quite different from each other, for each one had to make a file containing information about the job to be run. Indeed, one job description file had to be created for each job, since the run number had to be specified within it. The script to submit jobs directly to Condor was quite similar to the original perl script used for GT2, with only a few minor changes being necessary. The other one, for the reasons detailed above, had to be entirely different.

The executable to be run is also specified in the job description file. In this case, the executable given there is not in fact the software that analyzes the data, but a script that runs other scripts that set environment variables, manipulate files, and eventually finally run the software to analyze the data. In these scripts, the input file for the job is

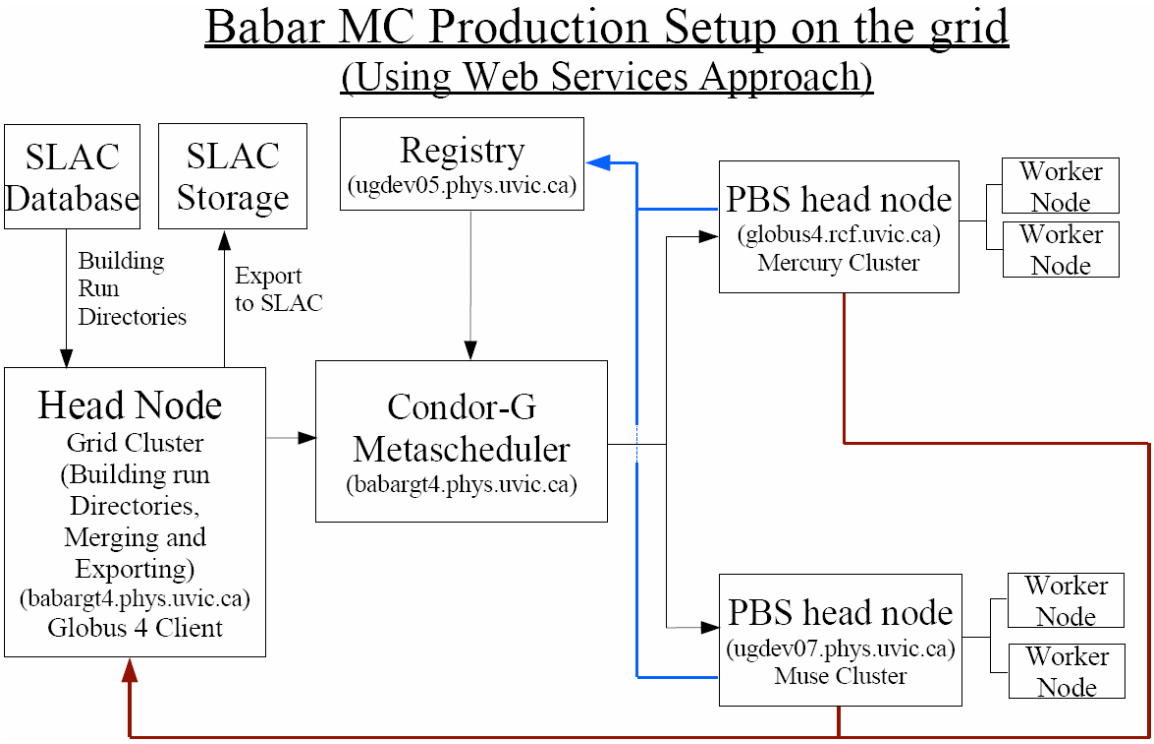
transferred, using Globus' file transfer service, to the directory where the job is to be run. Due to changes between GT2 and GT4, this required modification. The details of the problem are fairly technical, but it had to do with the fact that the job was executing as a different user than the one that submitted it. However, it was only a problem when the job was submitted through Globus to Condor, not when it was submitted directly to Condor, and the transferring of input files can be specified in the XML job description file syntax used by Globus, so this was used instead.

This was the only major problem encountered, and once it was resolved the efficiency of job submission with GT2 as compared with GT4 was tested. It was again thought that the GT4 job submission service might be less expeditious than the GT2 job submission service, since it is Web Services-based. To this end, many jobs were submitted to the GT4 metascheduler, both directly to Condor and through Globus to Condor. However, it was only possible to submit jobs directly to Condor on the GT2 metascheduler; submitting through Globus to Condor was not tried. Submitting directly to Condor does not take Globus completely out of the equation, however, since it is still involved in the metascheduling process. It was found that the average time from the job's submission to its ultimate completion when submitting directly to Condor was actually slightly lower when going through the GT4 metascheduler. Going through Globus to Condor was slower than either, as was expected due to the extra steps involved, but the time difference was, as in the file transfer case, practically negligible. The average times measured are as follows: submitting directly to Condor on the GT2 metascheduler took 274 minutes; submitting directly to Condor on the GT4 metascheduler took 273 minutes; and submitting through Globus to Condor on the GT4 metascheduler took 276 minutes. These numbers were generated by averaging the total elapsed times of about twenty runs.

4. Conclusion

There are two major areas of work still left to be done. Firstly, the monitoring of the running jobs on the existing grid needs to be improved. Some work was done on this, but much is left to be done. At present, there is a perl script that can be run that will generate a graph of the failed and finished jobs on the Muse cluster, but this script will have to be significantly improved upon and set up to run automatically. Secondly, it was

originally intended that the metascheduler have access to two major GT4-based clusters: the Muse cluster and the Mercury cluster. However, the Mercury cluster has not yet been set up such that the metascheduler can send jobs to it. Setting it up in this fashion should not be difficult; it should simply be a matter of mimicking the configuration of the existing cluster. Ultimately, the layout of the grid should be as presented in the following diagram:



5. Acknowledgements:

Thanks especially to Dr. Ashok Agarwal and Dr. Randall Sobie for their supervision and help throughout the term. Thanks also to Duncan Penfold-Brown, Dan Vanderster, Ian Gable, Ron Desmarais, Howard Peng, Greg King, and Andre Charbonneau..

6. References

[1] BaBar Project (2007)

http://www-public.slac.stanford.edu/babar/

[2] The Globus Toolkit (2007)

http://www.globus.org/toolkit/

[3] GridX1 (2007)

http://www.gridx1.ca/

[4] GT4 Release Manual (2007)

http://www.globus.org/toolkit/docs/4.0/

[5] GT2 Release Manual (2007)

http://www.globus.org/toolkit/releasenotes/2.0/

[6] Condor (2007)

http://www.cs.wisc.edu/condor/

[7] Condor Administrator's Manual (2007)

http://www.cs.wisc.edu/condor/manual/v6.8/3_1Introduction.html#6338

[8] PBS (2007)

http://www.openpbs.org/