University of Victoria
Faculty of Engineering
Spring 2003 Work Term Report

Extending Grid Computing in Canada: Federating Grid Canada and LCG

Department of Physics and Astronomy
University of Victoria
Victoria, Canada

Alexandros Dimopoulos
0027220
Electrical Engineering
adimopou@engr.uvic.ca

April 2003

In Partial Fulfillments of the Requirements of the B.Eng. Degree

Supervisor's Approval: To be completed by Co-op Employer

I approve the release of this report to the university of Victoria for evaluation purposes only

This report is to be considered ☐NOT CONFIDENTIAL ☐CONFIDENTIAL

Signature	Position	Date
Name(print)	email	Fax#

Abstract

New high energy physics experiments are requiring computing resources many times larger than those in existence. To meet their needs, they are turning to an emerging technology called computational grids which link many distributed computers together. Many groups around the world are actively deploying grids on regional, national, and international scales. If several grids are linked or "federated", then a larger computational power can be unleashed. An attempt was made to federate two dissimilar grids: LCG and Grid Canada using a centralized resource broker and job queue with promising results.

Contents

List of Figures

1 Introduction

The standard model has been one of the most successful theories of modern physics. It describes how fundamental particles are composed, as well as many of their interactions. However, several important questions about the nature of matter remain unanswered, such as what causes mass? It is theorized that mass is due to interactions with a field which permeates space called the Higgs field, and its associated particle, the Higgs boson [1].

For the past 12 years [2], some 2000 physicists from 34 countries have been collaborating to create the ATLAS detector which will begin searching for the elusive Higgs boson in 2007. The ATLAS experiment will use the new Large Hadron Collider (LHC) located at the CERN laboratory in Geneva, Switzerland. ATLAS is one of the largest, most expensive undertakings in scientific history and has been pushing technological boundaries since its inception.

One of these boundaries is computation. ATLAS will collect an enormous amount of data, an estimated 12-14 PB per year. The analysis of this data will require some 70 000 modern day equivalent CPUs [3]. Even when taking Moore's law into account, ATLAS will require huge computational resources while it is running. Additionally, the data, as well as these computational resources will have to be accessible to thousands of researchers spread all over the world. No existing computing facility can meet these requirements, and building one would be prohibitively expensive. Instead, ATLAS will use a new paradigm, the computational grid, to construct a virtual super computer. This will be done by seamlessly linking the computing facilities of the various organizations taking part in ATLAS; this is no small feat, as they are spread over the globe and are administered according to different sets of rules.

Across the world, there are regional efforts to build local computing grids, among them is Grid Canada. CERN is also constructing a grid, the Large hadron collider Computing Grid (LCG). Theses various grids are designed and implemented along different principles and standards, and are not necessarily compatible. In order to utilize these computing resources to their maximum capacity, various grids will have to be linked, or "federated".

This report briefly describes the generic grid architecture, the structure of Grid Canada, and an effort to federate Grid Canada and LCG.

2 discussion

2.1 The Generic Grid Architecture

The computational grid derives its name from the electrical power grid, which it resembles on a conceptual level. In both structures, resource consumers (of electrical power, or of computer cycles) are linked transparently across geographical and administrative domains to resource providers. In a power grid, generators in British Columbia and Oregon may power the lights for someone living in San Francisco. In a computational grid, a physicist at the University of Toronto may analyze data stored in the UK using computers in Japan and France. In both cases, the only resource the consumer sees is a nebulous structure called the grid. There no need to know where the resources are actually coming from to be able to exploit them. And in both cases, the resource providers are distributed geographically and administer their facilities according to different rules (such as government regulations for the power utilities and security settings for the computer clusters).

At the core of the grid is the concept of the virtual organization (VO). Unlike conventional networks, such as the world wide web, grids revolve around sharing actual computing infrastructure, not just information. Such sharing requires strict controls over who gets to use what, when, and how. A set of users who agree upon, and abide by a set of rules governing this sharing is called a VO. For grids to be powerful, they need to be easy to implement and use, this means that VOs must be scalable and be able to include anyone. This leads to a design modeled on a hourglass shape. At the top are the many different applications for the grid, at the bottom are the various infrastructures on which the grid is built, and in the middle are the small number of protocols and services which link the two ends.

Using this hourglass model, the grid architecture has four layers: fabric, connectivity, resource, collective, and application.

The fabric layer, as its name suggests, is the lowest layer. It consists of the actual resources upon which the grid is built, such as computational, storage, and network facilities. Elements composing this layer should have the capability to report their structure, state, and capabilities. This layer forms the bottom of the hourglass.

The connectivity layer enables grid specific network transactions. This layer has two parts. The first concerns connecting various fabric elements over networks, this can be accomplished with the existing TCP/IP protocols. The second is authentication, the identities of users and resources need to be verified in a secure fashion. Significantly, the authentication should provide users with single sign on and delegation capabilities. Single sign on allows users to log in to the entire grid just once per session without any further need to manually authenticate with individual fabric elements. Delegation allows a user's application to run on the grid on that user's behalf.

The resource layer consists of protocols, APIs and SDKs which govern the secure initiation, monitoring, control, accounting, and payment of jobs on individual resources. This layer is only concerned with connecting to single fabric elements, not with the bigger picture. Together, the connectivity and resource layers form the neck of the hourglass.

The collective layer forms the beginning of the top of the hourglass. This layer does not concern any one fabric element, but rather a collection of these, and is where the grid takes form. This layer implements services like directories which allow VO members to discover resources, and resource brokers which match jobs and grid resources.

Finally, at the top of the hourglass is the application layer. This layer consists of the actual user applications which run on the grid using the other four layers.

2.2 The Globus Toolkit

Over the past few years, the Globus Toolkit has become the de facto standard implementation of several of the grid layers described above. This set of software is produced by the Globus Alliance, a partnership between Argonne National Laboratory, the University of Southern California, the University of Chicago, the University of Edinburgh, and the Swedish Center for Parallel Computers.

The parts of the Globus Toolkit which are of main interest for this report are GSI and GRAM, the implementations of the connectivity and resource layers (the neck of the hourglass).

2.2.1 GSI

GSI (Grid Security Infrastructure) implements the authentication part of the connectivity layer (TCP/IP is used for actual connectivity) [6]. The GSI is based on public key cryptography and certificates encoded in the X.509 format. Every member of a grid, whether a user or a resource, receives a certificate. A certificate consists of a subject name which identifies the person or resource which holds it, the subject's public key, the identity of the signing certificate authority (CA), and the CA's digital signature. The CA is a third party which issues and guarantees the validity of certificates. It is a good idea for each VO to set up its own CA.

Suppose that Alice wants to simulate a proton collision over her grid, and her job gets sent to a linux cluster called Bob. Before Alice can use Bob, the two must mutually authenticate to ensure that they really are who they claim to be. First, both Alice and Bob need to have the CA's public key so that they can check the CA's signature on each other's certificates. Using SSL, Alice will send her certificate to Bob. Bob will first use its copy of the CA's public key to make sure that Alice's certificate is valid by checking the CA's signature on it. Next, Bob will generate a random message which is sent to Alice. Using her private key, Alice will encrypt the message and send it back to Bob. If Bob can get its message back with Alice's public key, then Alice's identity is verified. Alice would then verify Bob's identity in the same manner.

The GSI handles single sign on and delegation with proxy certificates. With the use of a password, a user can create such a proxy which is valid for a limited time. The proxy uses a slightly altered version of the owner's subject to identify it as a proxy certificate. A new set of keys is created and the owner signs the proxy. This time-limited proxy is used along with the user's certificate for authentication purposes, and the proxy is used to run jobs and access resources on the user's behalf. Thus after entering a single password, a user can access any resource on a grid to which he is entitled. In theory, a proxy could then create another proxy and thus a singe job could fork many children which could run all over the grid on behalf of the user. This however does not take place and will be discussed later.

2.2.2 GRAM

GRAM (Grid Resource Allocation Manager) is the Globus implementation of the resource layer [7]. GRAM simplifies job submission and control by providing a uniform interface to local job scheduling systems on grid computing elements. On the resource side, a job manager is implemented for the local job scheduling system (such as PBS or Condor). A job manager is a Perl module which defines a set of subroutines that wrap around the local job scheduler's interface, allowing GRAM to submit jobs to the local machine. This allows the user which is submitting jobs to be completely abstracted from all aspects of a computing element's local environment. GRAM does not provide any facilities for matching jobs to resources, this is taken care of in the collective layer which is not implemented here.

2.3 Description of a Couple of Grids

2.3.1 LCG

To handle the anticipated computing needs of various LHC experiments, such as ATLAS, CERN is overseeing the development of a global computational grid called the LHC Computing Grid (LCG). This grid is to handle the simulation, processing, and analysis of data collected for LHC experiments. This is a world wide project comprising of computing centers around the world. A major design feature of this grid is that all participating resources must be fully dedicated to LCG. In other words, resources may only run jobs from LCG and nothing else. This requirement makes it impossible for facilities which are shared by several research groups (such as the University of Victoria's Mercury cluster) to participate.

2.3.2 Grid Canada

Grid Canada is a partnership between CANARIE, the National Research Council (NRC), and C3.ca to build a Canada wide grid. The grid fabric is currently comprised of three computing facilities spread across Canada: Mercury at the University of Victoria, Thuner-gw at the University of Alberta, and Mercury at the NRC in Ottawa. The development effort is lead by Dr. Randall Sobie's group in the department of Physics and Astronomy at the University of Victoria. Grid Canada is following a design philosophy which is opposite to LCG's. This grid uses only shared facilities, not one is dedicated to only running Grid Canada jobs. The intent is to put as few requirements on participating sites as possible, and to essentially use up their spare resources.

2.3.3 Why Federate these Grids?

Since all of Grid Canada's resources are shared among different researchers, they cannot become part of LCG. However, Canadian ATLAS members would still like to use these shared facilities to run computing jobs pertinent to ATLAS. The problem is that these are run on LCG. The solution is to federate Grid Canada and LCG so that jobs submitted to LCG can be seamlessly transferred and run on Grid Canada.

2.4 The Grid Canada Resource Broker

A grid's resource broker (RB) is part of the collective layer in the architecture. It is the component which matches jobs to resources, acting as a middleman (or broker) between users and individual resources. Any scheme to federate grids has to be intimately connected to their resource brokers.

2.4.1 The Previous Resource Broker

The existing resource broker for Grid Canada had been implemented on a centralized model [8]. All jobs submitted to Grid Canada had to be passed to a single resource broker which then submitted them to resources. By default, GRAM client libraries are configured to only allow delegation of limited proxies. Such a proxy cannot re-delegate other proxies. The result is that a job can only be sent to a single resource using GRAM without the ability to migrate somewhere else. In this case, if jobs were sent to the resource broker using GRAM, they would be stuck there and never reach a resource. Since no way had been found to allow a user to delegate a full proxy, a workaround was made using a MySQL database. A user would enter his job and full proxy certificate file into a centralized MySQL database. The resource broker, which was running as a cron job on the machine where the database resided, would query a GIIS (Grid Index Information Service) server to find acceptable resources, then submit the job to one of them. A GIIS is a database where grid resources publish information such as their names, capabilities, and statuses. This system is illustrated below in figure 1.

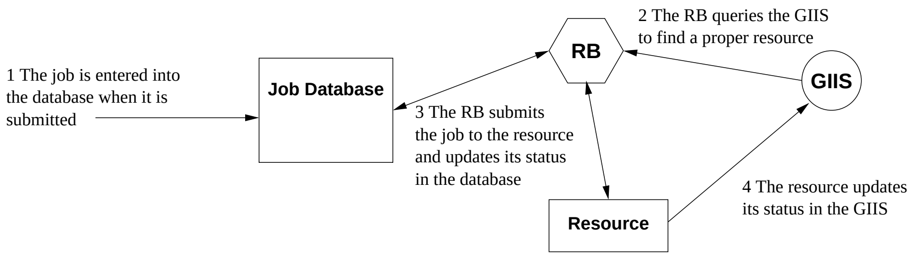
1 The job is entered into the database when it is submitted
Job Database
RB
2 The RB queries the GIIS to find a proper resource
GIIS
3 The RB submits the job to the resource and updates its status in the database
Resource
4 The resource updates its status in the GIIS

Figure 1: Job submission under the old Grid Canada resource broker.

2.4.2 The Need for a New Resource Broker

A major problem with the existing resource broker was its enormous security vulnerability. The entire PKI (public key infrastructure) used in the Globus Toolkit was circumvented with the MySQL database. Other than the fact that no mutual authentication was needed to access the database, the password to access it was in the Perl script which acted as the client submission tool. Furthermore, this password was the same one used by the resource broker, which means that anyone accessing the database with it could not only make new entries, but modify existing ones.

The specialized interface to the resource broker would have made it difficult to link easily with other grids. LCG would most likely have to gain access to Grid Canada through its resource broker in order to ensure that Grid Canada could maintain control over its resources. LCG (and most others) uses GRAM to communicate among elements, how would it talk to Grid Canada's RB?

The resource broker was not extensible enough so a new one had to be designed and implemented if Grid Canada and LCG were to be federated.

2.4.3 Designing a New Resource Broker

TRIUMF, located in Vancouver, is to be the main Canadian LCG site, thus it made sense to try to federate the two grids at this point. After consulting with some members of the Simon Fraser University (SFU) Department of Physics who are involved in setting up TRIUMF for LCG, an overall architecture was arrived at. TRIUMF was going have several regional computing facilities at its disposal for executing LCG jobs, so it was going to have its own resource broker. It was decided that this resource broker would send jobs to a new Grid Canada resource broker using GRAM. Grid Canada was to look as just another computing facility to TRIUMF, and TRIUMF was to appear as just another user to Grid Canada. This is illustrated in figure 2.

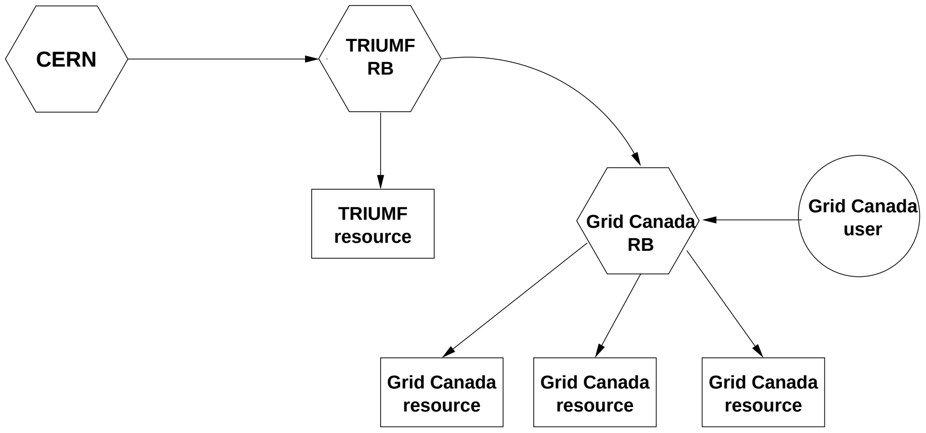

Figure 2: Job migration in the LCG-Grid Canada link.

The new Grid Canada broker could be centralized like the old one, or it could be decentralized. In a decentralized model, the job submission client is also the resource broker. It queues jobs, decides where to submit them, and then sends them off. The main advantage of this scheme is its robustness. Because each and every grid user has his or her own personal broker, the grid does not become paralyzed if one of them fails. However, it is very difficult to monitor the status of the entire grid in such a model. Because there is no centralized job queue, it is difficult to ascertain the total number of jobs submitted to the grid or their status. Conflicts between resource brokers would also arise. If two users with ten jobs between them simultaneously select the same resource which has the capacity to accept another six jobs, who has priority? Whose jobs get submitted first? How many of each user's jobs get submitted? How is all this decided?

If the centralized model was again chosen, then the system would still have a critical point of failure. However, complicated problems like inter-broker conflicts would be avoided. Such a system with a single queue would be easier to monitor. It would also be easier to update and modify since only one copy would have to be changed, instead of having to ensure that every user has the most up to date version of the software.

There was also a third option which was briefly considered: a centralized broker with reserved ticketing. In such a system, users contact a centralized broker to request resources for their jobs. When the broker finds a suitable facility, it contacts it and reserves certain resources on the user's behalf. The broker then sends a ticket to the user which specifies the reserved resources. Using this ticket, the user then claims his reservation and submits his jobs. Like the centralized model, this allows for easy monitoring, but it is very scalable since the resource broker does not manage a job queue. Implementing a ticketing system is not trivial as there are major security issues to address.

In the end, the centralized model was chosen. It could adequately handle the work load since Grid Canada is not very large. It was also the most straightforward to implement.

2.4.4 Building the New Resource Broker

One option was to essentially reimplement the existing resource broker so that it would have a GRAM jobmanager. When submitting a job, a user would send it with GRAM to this jobmanager. The jobmanager would then insert the job into the database and the resource broker would then take care of the rest.

A more attractive solution was put forward by one of the SFU physicists, Dr. Rodney Walker. He was implementing TRIUMF's RB, and planned to use Condor-G to do so. Condor [9] is a powerful batch system which has been developed over the past 15 years at the University of Wisconsin-Madison's Department of Computer Science. A very attractive feature of Condor is its very flexible mechanism for matching jobs to resources. Condor-G is Condor extended to use the Globus Toolkit. Condor can manage jobs only on pools of local machines, say within a university. Condor-G is intended to extend these pools to include remote resources using GRAM. Condor provided a sophisticated queuing system and a resource broker with its matchmaking system. All that was needed was a jobmanager and some nonconventinal configuring of Condor.

The jobmanager was relatively easy to develop. There already existed a regular Condor jobmanager to allow jobs to be submitted to a resource's condor queue. With a few modifications, this jobmanager could be used for our purposes.

A major obstacle to implementing a Condor-G solution was the handling of user proxies. The intended submission path for Condor-G uses only one GRAM link. A job is first submitted locally to the Condor queue. Next, Condor-G submits the job to a remote resource using the user's full proxy for authentication. In order to be useful as a grid resource broker, Condor-G would have to support two GRAM links, once from a user to the RB and again from the RB to the resource. As things were, a user would use his full proxy to authenticate with the RB machine, and send his job to the Condor-G jobmanager with a limited proxy. The job would then sit in the queue indefinitely because the limited proxy can not be used to authenticate with any resources. A way had to be found to send a full proxy to the resource broker.

After searching online mailing lists, a way was found to modify the GRAM client libraries so that users could delegate full proxies. This solution was however far from ideal. First of all, once this change was made, users could no longer delegate limited proxies. Also, this modification would have to be applied by anyone who would submit jobs directly to Grid Canada. Secondly, this fix would not work for the link with TRIUMF since no jobs there would have full proxies. Using this would have entailed changing the way LCG operates which was not an option.

The answer was to use MyProxy [10], an online credential repository system. This system allows users to store their credentials on a server and to later retrieve them. It is possible to store one's full proxy in a repository and to later retrieve it using only a limited proxy. The Condor-G jobmanager was further modified to retrieve a user's full proxy before submitting his job to the queue so that Condor-G could then submit the job on the user's behalf to a resource. The only requirement was that users (both Grid Canada and LCG) store their proxies in one of CERN's or Grid Canada's MyProxy servers before submitting a job.

As mentioned above, Condor has built in facilities to match jobs and resources. Condor's matchmaking system is based on something called ClassAds. A ClassAd is analogous to the ads in a newspaper's classified section. Every job has its own ClassAd which lists its requirements such as required memory, clock speed, maximum acceptable load on a machine, and etc. Every resource also has its own ClassAd listing its attributes. Condor reads all the ClassAds and uses the information in them to match jobs to resources. The job ClassAds are usually written by the submitter, but to simplify things with the RB, they are written by the jobmanager and are all identical to each other. Ideally, ClassAds for the Grid Canada resources would be generated using information in the Grid Canada GIIS and advertised to the Grid Canada RB. A cumulative ClassAd for the entire grid which would represent it as a single resource would also be generated and sent to the TRIUMF RB. The problem was that the information published in the Grid Canada GIIS was not particularly useful for this purpose. Nor was it in a format compatible with what LCG uses. Changing the GIIS was resulting in unforeseen complications. So to get a working system up and running as quickly as possible, a set of Perl and shell scripts were written to generate the resource and grid ClassAds.

2.4.5 Results

The ClassAd scripts worked well. Condor at both at Grid Canada and TRIUMF read them and accepted the resources they described.

In limited tests, simple jobs were able to pass from the TRIUMF RB, to the Grid Canada RB, and to a Grid Canada resource. Unfortunately, the system began to fail. Jobs would be sent the Grid Canada RB, get matched to a resource, but never be sent there. Extensive debugging and testing to resolve this problem was not done before the writing of this report due to time limitations.

3 Conclusions

Although the Grid Canada Condor-based resource broker was not yet fully functional, it showed the ability to link two computational grids.

4 Recommendations

The Grid Canada GIIS should be modified so that its information can be used to generate resource ClassAds. The problem causing jobs to remain in the queue after they have been matched to a resource needs to be resolved so that the Grid Canada resource broker can be extensively tested for stability. Finally, it may be worthwhile to develop a reserved ticketing based resource broker to allow for greater scalability.

References

[1] M. Barnett, ”The ATLAS Experiment,” [Online Document], Available at HTTP: http://atlasexperiment.org/ index.html

[2] ”ATLAS at Victoria: The Energy Frontier,” [Online Document], Available at HTTP: http:// particle.phys.uvic.ca/ web- atlas/ atlas/ overview/

[3] W. Tomlin,”LHC Computing Grid Project (LCG) Home Page,” [Online Document], 2003 Dec 10, Available at HTTP: http://lcg.web.cern.ch/LCG/

[4] I. Foster, C. Kesselman, S. Tuecke, ”The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” The International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200-222, 2001.

[5] Globus Webmaster, The Globus Alliance, [Online Document], 2004 April 27, Available at HTTP: http://www.globus.org

[6] Globus Webmaster, ”Overview of the Grid Security Infrastructure (GSI),” [Online Document], 2003 May 19, Available at HTTP: http://www-unix.globus.org/security/overview.html

[7] Globus Webmaster, ”Resource Managements,” [Online Document], 2004 March 31, Available at HTTP: http://www-unix.globus.org/developer/resource-management.html

[8] L. Klektau, ”Running ATLAS Experiment Simulations on a Grid Environment,” [Online Document], 2004 Jan 4, Available at HTTP: http://grid.phys.uvic.ca/docs/report-lila.pdf

[9] Condor-Admin, ”Condor Project Homepage,” [Online Document], 2004 April, Available at HTTP: http://www.cs.wisc.edu/condor/

[10] J. Basney, S.S. Chetan, J. Gawor, D. Kouril, J. Novotny, M. Ruda, B. Temko, V. Welch, ”MyProxy Online Credential Repository,” [Online Document], Available at HTTP: http://grid.ncsa.uiuc.edu/myproxy/