University of Victoria
Faculty of Engineering
Fall 2007 Work Term Report

Xen-based Grid Computing Cluster and Condor SOAP Client

Department of Physics
University of Victoria
Victoria, British Columbia

David Gong
0436353
Computer Engineering
daobgong@engr.uvic.ca

January 3, 2008

in partial fulfillment 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.

The report is to be considered ☐ NOT CONFIDENTIAL ☐ CONFIDENTIAL (select one)

[redacted]

Signature | Position | Date

[redacted]

Name (print) | E-Mail | Fax #

If a report is deemed CONFIDENTIAL, a non-disclosure form signed by an evaluator will be faxed to the employer. The report will be destroyed following evaluation. If the report is NOT CONFIDENTIAL, it will be returned to the student following evaluation.

305-4941 Lougheed Hwy
Burnaby, British Columbia
V5B 4S6

Mr. Roel Hurkens
Co-op Coordinator
Faculty of Engineering Hurkens
University of Victoria
P.O. Box 1700
Victoria, B.C.
V8W 2Y2

December 26, 2007

Dear Mr. Hurkens,

Please accept the accompanying Work Term Report entitled "Xen-based Grid Computing Cluster and Condor SOAP Client"

This report is the result of work completed at the University of Victoria, Department of Physics. During my first work term as an engineering student at the University of Victoria, I was engaged in setting up the Xen-based virtual worker node in computing grid, and evaluating its performance in comparison with the native nodes. I also participated in the project of developing job submission clients through the Condor SOAP API, which facilitates the remote job submission to the computing grid.

Through the course of the term, I was given the opportunity to learn much about building computing grid, integrating software, and Linux system administration. The experiences I gained during working on these projects will be very helpful in building a future career.

I would like to extend my thanks to Dr. Randall Sobie, Dr. Ashok Agarwal, Dan Vanderster, Ian Gable, Howard Peng for their support and help.

Sincerely,

David Gong

LIST OF TABLES AND FIGURES

FIGURES Page

TABLES

Summary

Modern scientific applications such as High Energy Physics demand very high computing power, which cannot be fulfilled by individual work stations. Computing Grids are created to achieve this work. Computing Grids consist of many computers with different hardware and software. Some have specific operating system requirements, such as Babar. To satisfy these requirements, it is usually preferred to convert one type of operating system and system software into another with minimum work. Virtual Machine Monitor such as Xen was introduced for this purpose. With Xen, we were able to create virtual worker nodes on a physical worker node. The virtual worker nodes appeared to be extra nodes. They are part of the cluster. With necessary software installed, those virtual nodes successfully ran HEP application, such as Babar jobs.
Based on the premium performance data, it shows Xen-based Virtual Machine gives acceptable performance. Further testing is still going on.
Condor was configured as a metascheduler. Job Submission was done traditionally by logging to the metascheduler through SSH. With the recent introduce of Globus Toolkit 4(GT4) using Web Services, Job Submission has become much more flexible than before. Condor SOAP API was created in order to make job submission to the Condor Pool and job control possible through Web Services. This project is aimed to develop a thin Job Submission Client through Condor SOAP API. Although Condor SOAP API is designed to facilitate job submission to Condor Pool instead of Computing Grid, the similarities between these submissions suggests that it is worth to try.
Using Java programming language, submitting jobs to condor pool and job control has been successfully tested. Authentication using Secure Socket Layer (SSL) has also been tested. However, job submission to GT4 grid has not been tested yet due to time constraint.

1. The Introduction

The University of Victoria Department of Physics and Astronomy has been doing Grid Computing research for quite a few years. As the increasing need of computing power and resource in many different fields in the modern world, Grid Computing becomes more and more popular.

Computing Grid is a way of aggregating computing resources (including processing, memory, and disk storage) to provide a platform for sharing. It is useful especially when it is impossible for a single computer to handle the task. Computing Grids often use the resources from scientific research institutes and universities.

Depends on the job, the software runs on Computing Grids sometimes has very specific needs on Operating System or other software support. Since Computing Grids often consist of very different hardware and operating system, in order to transfer the heterogeneous environment into a homogeneous environment to fulfill the needs of the specific needs of the grid application software, Virtual Machines make it possible. Due to the size and complexity of the application software, it is often easier to choose an operating system that suits the needs of the application instead of the other way. Xen becomes a good candidate since it is known to offer near native performance [1].

Performance can be affected by many factors such as the hardware enviroment and the operating system; it also depends on the application. We are particularly interested in how Xen performs while it is running High Energy Physics application, such as BaBar.

Traditionally, job submission to computing grid is done by physically accessing to or remote logging into a submit host (usually a metascheduler). With the emerging Web Services, more flexible methods make job submission from any host possible through the installation of certain software, such as the GT4. However, methods like a thin Job Submission Client (easy installation) or Web Submission (no installation) are more attractive to users, especially to those who only occasionally use grids.

With the recent release of Condor SOAP API, web submission and a thin Job Submission Client to Condor Pool are possible. Problems need to be overcome in order to use this API to submit jobs to a computing grid.

1.1 Basic Components and Services of a Computing Grid

1.1.1 Submitting Host

A submitting host is a machine that a user uses it to submit jobs to computing grid through metascheduler. Usually it is locally or remote accessible to the user, and has necessary software such as Globus Toolkit installed. Metatscheduler is also frequently directly used as submitting host.

1.1.2 Metascheduler

a metascheduler is the gateway to a grid. Since a Computing Grid usually consists of many clusters at different geographical locations. A common gateway to all the available cluster resources can make submitting jobs to many locations easy and efficient by balancing the workload between clusters. Usually there is a local resource scheduler at each cluster; the metascheduler is actually a scheduler of many local schedulers.

1.1.3 Cluster

A Cluster is a cluster of many computers. All the computers in the cluster are called Worker Nodes except for one computer called Head Node that is responsible for communicating to the world outside. All the Worker Nodes are available to do real computing work. Since Head Node is the only gateway to the cluster, there is a lot of network traffic a Head Node has to handle. Opening only one gateway makes the cluster safe and simplifies the network administration work.

1.1.4 Central Registry

Central Registry is responsible for collecting all the resource information from many clusters. Metaschedulers polls resource information from central registry when it is needed. This liberates the metaschedulers from maintaining big resource information database and metaschedulers can focus more on handling client job flow.

1.1.5 Globus Toolkit

Globus Toolkit is usually installed on all nodes that form the computing grid. Many grid services such as security, data management information services etc. are provided by Globus Toolkit. Without it, the file transfer is not possible, authentication is also complicated. [2]

1.1.6 Condor

Condor is usually installed on metascheduler to do job and resource management. Condor is developed by the University of Wisconsin. It is software that supports High Throughput Computing on large collection of distribut owned computing resources [3]. Because it has the management capability on distributed resources, it can used as metascheduler of the computing grid.

1.1.7 Portable Batch System (PBS)

Portable Batch System is usually installed on all the nodes that form a cluster. PBS performs job scheduling. It allocates computing jobs to worker nodes in the cluster. When it is used as local resource manager, it receives jobs from metascheduler and assign jobs to worker nodes. Result can be sent to metascheduler, or directly to destination.

1.1.8 Authentication

To prevent misuse of computing resource, only authorized users should be allowed to use. A Computing Grid uses many authentication methods such as File System, Kerbros, Grid Security Infrastructure, and Secure Socket Layer etc. Often x509 user certificate is chosen to identify users.

1.1.9 Proxy and Delegation

Computing jobs frequently require many hours to complete; resource authorization is required during the period. It is not possible for users to wait in front of the computer to enter password to identify them. A proxy consists of slightly modified, time limited x509 certificate and password-less private key is generated from the original certificate and private key [5]. This proxy is deposited in the Myproxy Server, it can be obtained whenever it is required.

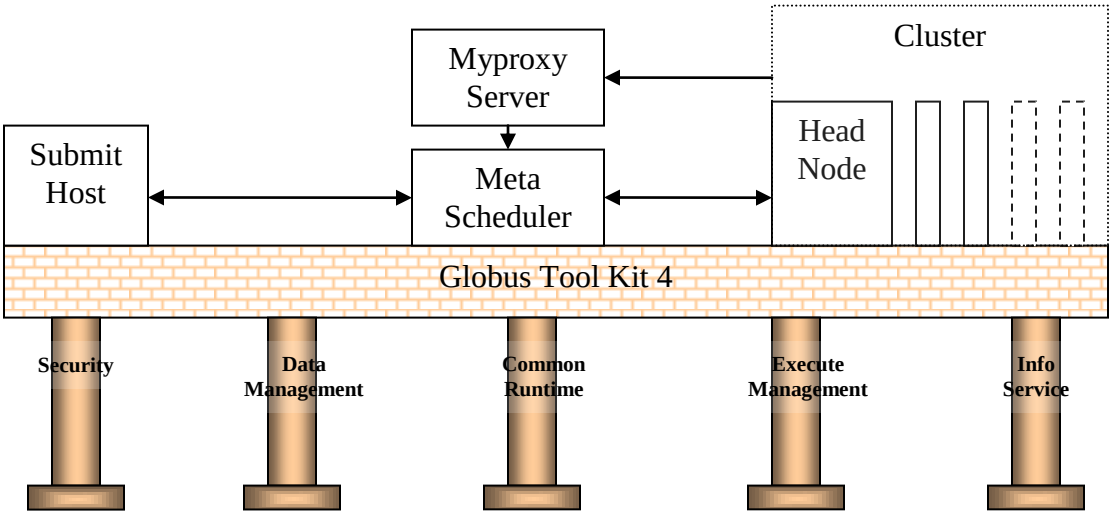

Figure 1.1 Basic Component of a Computing Grid

1.2 Motivation of the project

In a grid environment, there are many reasons that require the computing nodes having certain Operating System and other software support. For example, most of the time, High Energy Physics (HEP) application is highly Operating System (OS) dependent [4]. It is not possible to use the many CPUs free at other universities and institutions who want to be part of a grid. Xen, the Virtual Machine Monitor, is found to be very useful to provide the desired OS through creating Virtual Machines with acceptable performance and minimum effort. By creating virtual

worker nodes on top of Xen should be able to provide the needed environment to run HEP applications.

This project is aimed to create Xen-based Grid on three different locations, the local muse cluster, mercury cluster, and some nodes in NRC, and test run HEP application such as Babar to evaluate the performance.

Job Submission to a Grid traditionally needs a user either having physical access to the metascheduler or remote SSH to it. This requires the user have a Kerberos account in the domain. It is not flexible. Another way of submitting job is install GT4, the installation and configuration of GT4 makes the Job Submission not a simple task, especially for people who only occasionally use it.

Web Service has been used in many fields of computing to make the service smart and system administration easy. As Condor implements SOAP API, a thin Job Submission Client is possible. Although GT4 also uses web service, it is not a small package. A thin Job Submission Client can either be easily installed on client's computer, or it can be put on web server for user to use without real installation.

2. Discussion

2.1 Xen based Grid

2.1.1 About Xen

Xen is microkernel where you can run multiple operating systems at the same time. This thin kernel handles switching between the operating systems and controls hardware access. It is similar to other Virtual Machine Monitors, but different from them by using a technology call para-virtualization where both kernels in the host and guests need to be modified to achieve good performance [1]. Due to the requirement of kernel modification, only Linux is possible to run as para-virtualized host and guests. With the recent CPU hardware support from both Intel and AMD, full-virtualization is possible and Windows can run as guests without modification. One significant advantage of Xen is that para-virtualization provides close to native performance which is one of the most important factor while choosing virtual machine monitors.

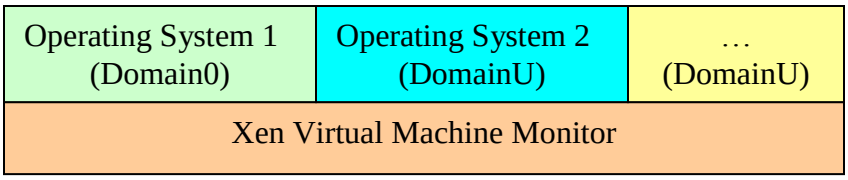

Figure 2.1 Multiple OS on Xen

2.1.2 Xen Version Consideration

Xen uses para-virtualization which requires Linux Kernel needs to be modified. Thus each Xen version comes with different version of modified Linux kernel. In the case that software needs specific Linux kernel version to support, the corresponding Xen version needs to be found. Luckily, most of the time, we only need to worry the major version of the Linux kernel version, not the minor update.

For example, Xen 2.07 gives Linux 2.4.30 and Linux 2.6.11.12, and Xen 3.02 gives only Linux 2.6.16

2.1.3 Guest Performance Factors

Memory

The more memory is allocated to a domain, the better the performance is expected. Since the total amount of physical memory is fixed, the memory will be shared by the host domain and guest domain. The following memory sharing strategies can be considered:

1. Fix the amount of memory in both guest and host domain

   This is the easiest way of implementation. But when both domains are not always busy, let's say the guest domain is idle, the host domain will not use the portion of memory allocated to the guest domain. Obviously the memory utilization is low. Or if a job requires the amount of

memory slightly exceeds the amount allocated to the domain and still far less than the total amount of the physical memory, the job won't be able to run on this domain.

2. Both the guest domain and host domain allocate minimum amount of memory at the beginning. When a job is submitted to a domain, all the extra memory detained by Xen will be available for this domain. A balloon command will be executed to expand the memory for this domain. The memory will be ballooned to minimum again after the job is done. This method basically only allows one of the domain execute job, but each domain can handle jobs that requires more memory. In the case that all jobs are memory demanding, and there is no enough memory to run jobs in both guest domain and host domain at the same time, this method is effective.
3. Another way of allocating memory is slight different from the previous one. Both the guest domain and host domain allocate roughly half the memory and allow to balloon down its memory to minimum and balloon up to maximum. this method is the most flexible one among all the methods.

Both method 2 & 3 requires memory ballooning. Since the ballooning command can only be issued through the hosting domain, it requires the host domain has the knowledge of what the guest domain needs. If the guests are started by the host domain on demand, then it is not a problem, since the host can easily get the requirements and try to fulfill it. If the guests are statically started, then a guest domain needs a mechanism to send it memory requirement to its host domain and check the response.

CPU

With Xen 2.x or lower, it is not possible to assign multiple CPUs to a single guest domain [6]. With the new version from Xen3.0 onwards, the user has the total control of assigning CPUs to domains. CPUs are assigned at creating time, and will not be able to change through the life time of the domain. With emulating more number of virtual CPUs than the actual number of physical CPUs assigned is not recommended due to performance reason.

In our muse cluster, each muse node has two CPUs. When Xen2.07 is enabled, Xen automatically assigned one physical CPU to domain0, and the other CPU is assigned to the guest domain when it is created.

Disk

Disk is one of the very important performance factors.

A guest domain file system can exist in one of the 4 places:

- Image file

  This is the easiest way of installing a guest domain just by putting the guest domain file system in an image file which exists on the host domain file system. An access to a guest domain file will need the guest domain file system manager's involvement first, the relevant parameters will be passed to the host domain file system manager, and it needs to readjust the parameters since the guest domain does not know it only exists in an image file. After the adjustment, the request will be sent to disk. Besides the extra step of readjustment of calculation, the I/O request still needs to compete with other partitions for the disk and other disks for bus.

- Partition

  Putting a guest domain in a partition is a good choice if possible. It requires free partitions and usually it is not easy to obtain without planning ahead while installing the host domain. But if the host domain is using Logical Volumes and has enough space, it is very easy to create extra Logical Volumes. Thus it is highly recommended to use Logical Volume in the guest domain. There are also many other advantages such as changing size of logical volumes.

  With a guest domain file system in a partition, a file access will be easier without the need of involvement of the host domain file system manager. Although the competition

from other partitions within the same disk and other disk's competition for bus still exist, the performance is expected to be much better than the image file case.

- Disk

  Putting a guest domain in a disk is the best choice provided there is a disk available. Since there is no competition from other partitions, the only competition will be bus; this gives the best disk performance.

- Network Mounted Files

  This might make the duplication and backup much easier, but this will really put a lot of traffic on the network. And the performance for this case is also very much depends on the traffic on the network. It is excellent for testing and temporary use, but not a good choice for permanent setting.

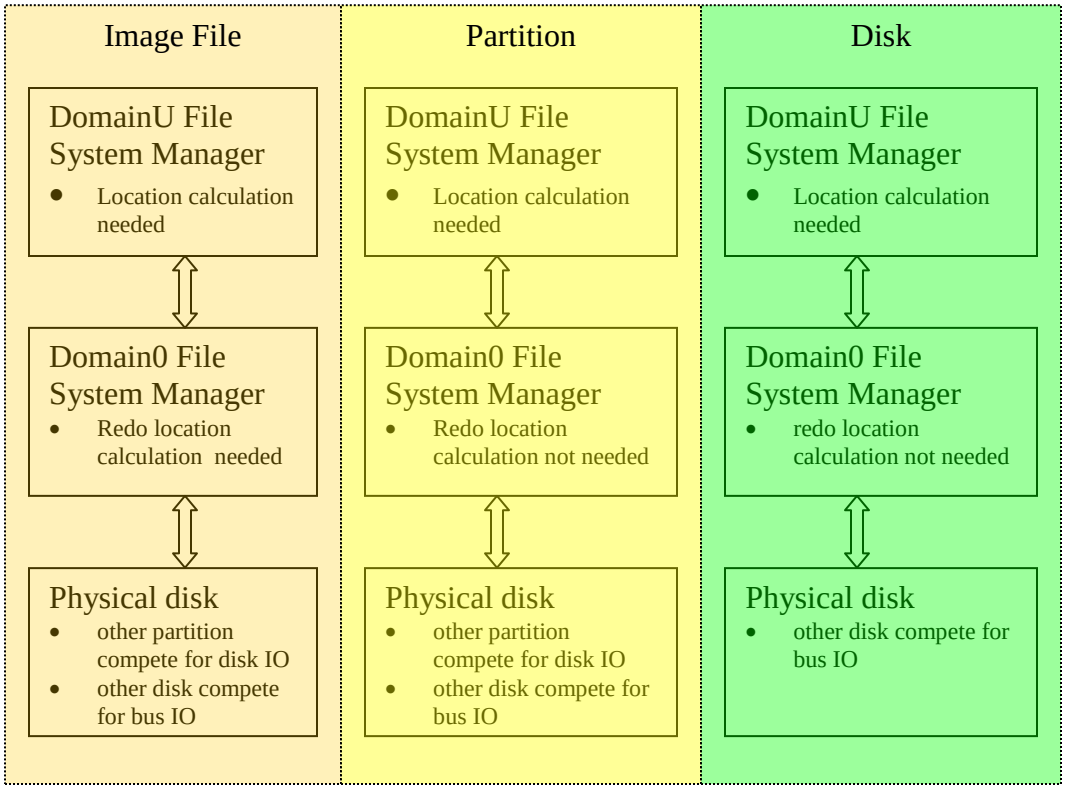

Figure 2.2 Guest File System Comparisons

2.1.4 Xen Node Networking Consideration

Bridged model

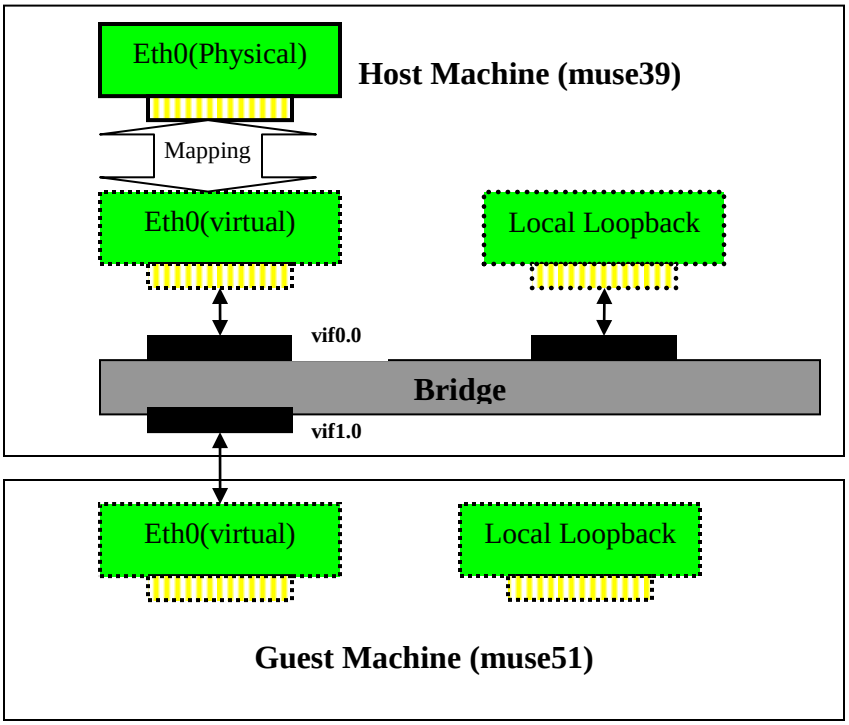

Figure 2.3 Networking - Bridge Model [7]

Bridging method allows each guest appears on the network as individual hosts through the software bridge in domain0. In this way, domainUs can use external DHCP service to get IP addresses, and this makes the network administration relatively easy. In our case, we chose bridging. Guests obtain IP address from DHCP server, but the IP addresses are fixed IP addresses.

Routed Model:

It creates a link between a Virtual Interface in domain0 to the Virtual Ethernet Card in each domainU [7]. This makes each guest domain can communicate with domain0. Routing entries are also added in the domain0 routing table at the same time. In this case, domainUs need static IP address. Without a static IP address, there is no way for domain0 to add the routing entries properly.

The disadvantage is that DHCP can not be used, since before the routing table was set up properly in domain0, the DHCP requires won't be able to reach the DHCP server, and the DHCP response also won't be able to reach the guest domain. And in order to set up the route, IP address is needed. This is a deadlock.

It is still possible to set up DHCP server on Domain0 with modifying the routing script [7]. It is not particularly useful unless we are creating many guest domains in a single machine. In our case, we are more interested in creating just one guest per machine.

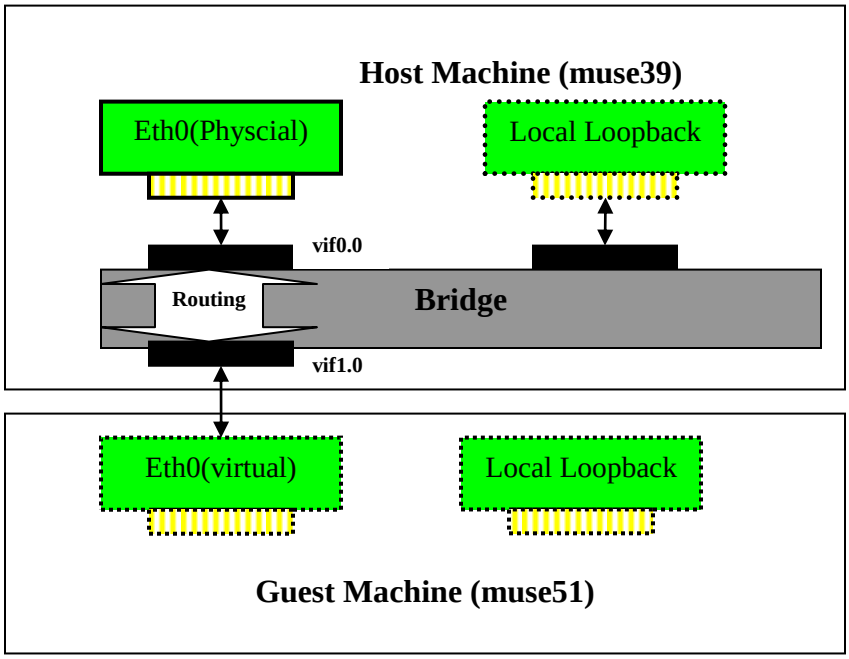

Figure 2.4 Networking - Routed Model [7]

2.1.5 Guest File System Preparation

There are different ways of preparing the guest file system. One script called bootstrap is particularly interesting because it only installs the very basic RPMs, and thus the installation is very straight forward and fast [8]. But due to the fact that a very minimum installation tends to

make the configuration of other services complicate and it is not recommended here. The recommended way is make a clean installation (close to full, just exclude X) on a machine, and port the files to the guest machine. It proves to be more reliable and more efficient.
The following services need to be disabled

Service	**Reason**
TLS	Xen does not support TLS because of bad performance
Time synchronisation	let domain0 handle
Yum auto update	auto update can make trouble

Table 2.1 Services to be disabled

2.1.6 Local Resource Manager – Portable Batch System

Portable Batch System is a distributed workload management system. It offers great flexibility usually by employing queuing system, scheduling system, and monitoring system [9]. It is very often used as a backend Local Resource Management system for Grid Clusters.

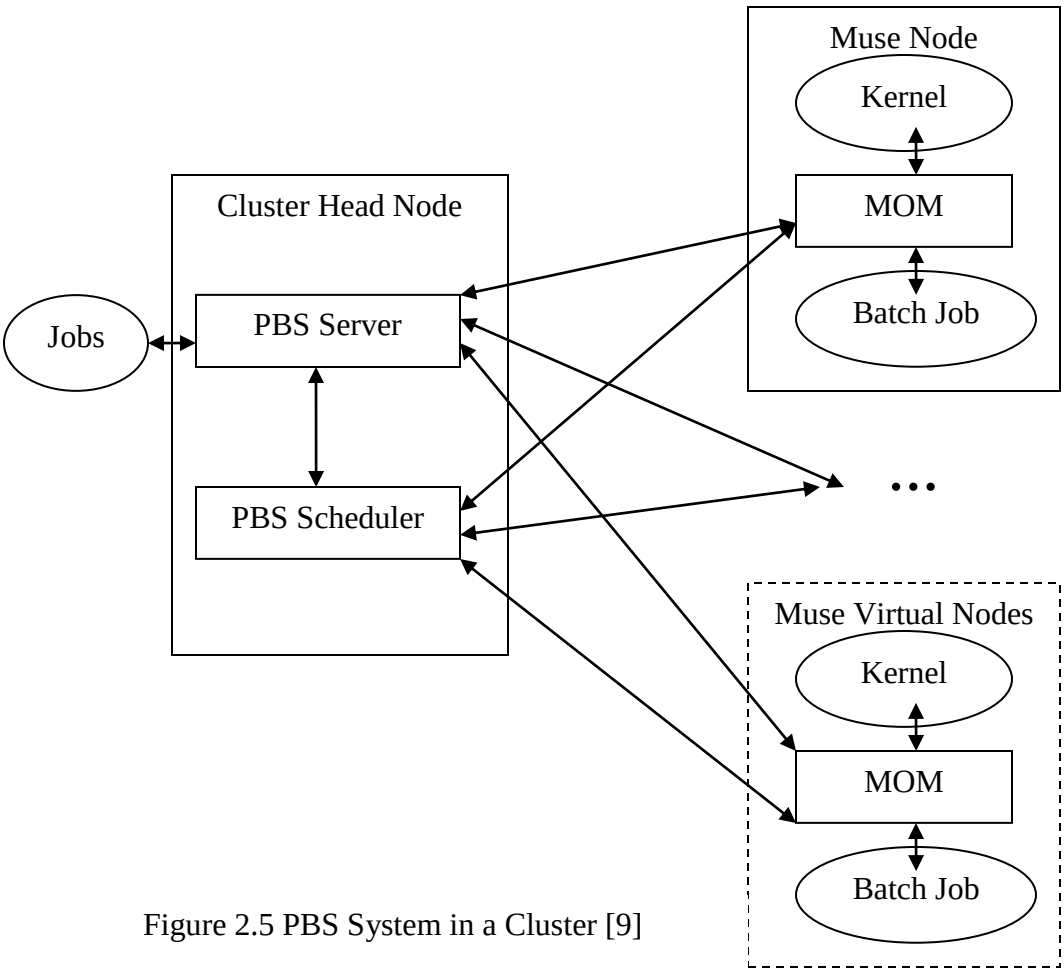

Figure 2.5 PBS System in a Cluster [9]

2.1.6.1 Job Routing Strategy in queues [3]

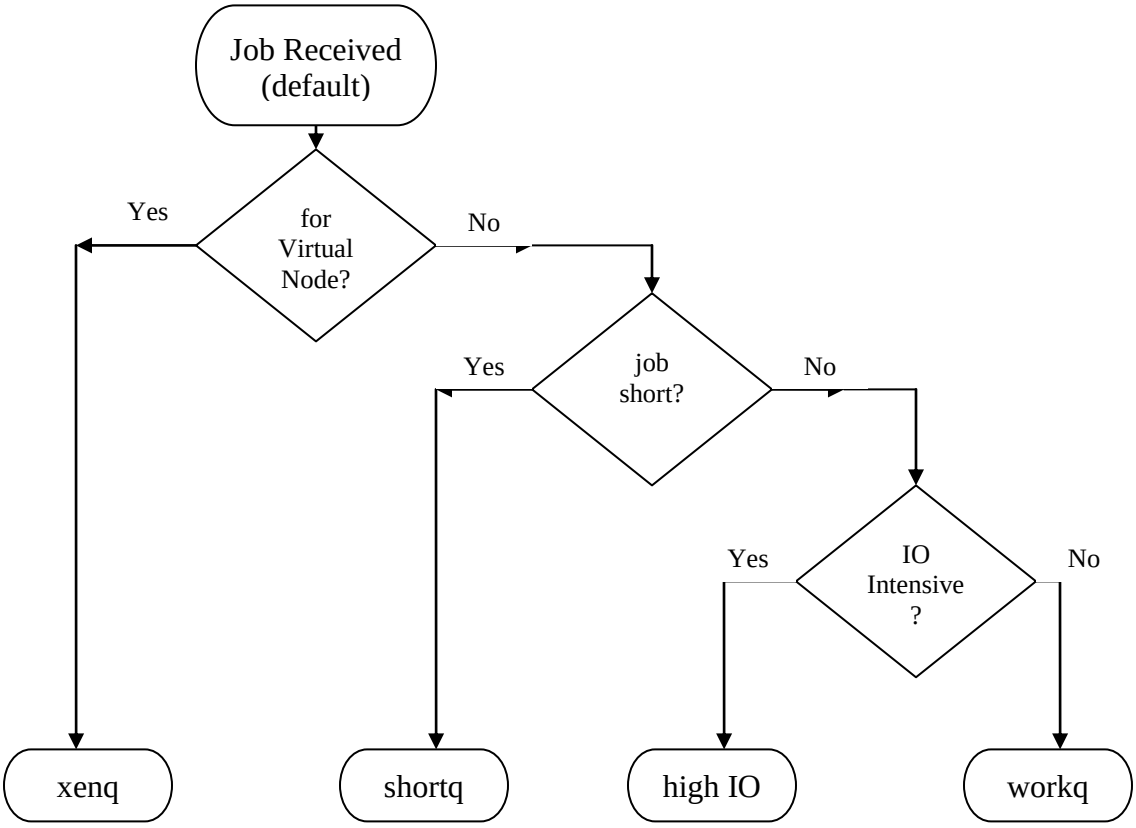

Figure 2.6 Job Routing

- Jobs are evaluated through the queue configuration file in order to different queues on PBS Server; jobs will be routed to the queue at the first match depending on the requirement of the job [10].
- Queue order in the queue configuration file is important to ensure the proper routing. It is recommended to put the queue with the most restrict requirement first. A job can match with one queue can also matches with queues have less restrictive resource requirement, thus frequently a job can matches with multiple queues. If the sequence is defined as from most restrictive to less restrictive, a proper routing is ensured [10].
- A test on a job resource requirement will pass if that requirement is not set. Thus it is recommended to have server wide or queue wide default values.

2.1.7 Globus Tookit 4

Globus Tookit 4(GT4) provides the infrastructure of a Computing Grid. It includes Security, Data Management, Execution Management, Information Service, Common Runtime services. These services are essential to computing grid and otherwise many software are needed to work together to make a Computing Grid work. Since it uses web services, it also makes the network traffic across site boundary easily and much simplifies the system administration work. GT4 is an open source software, but it has become the defector standard of the Computing Grid.

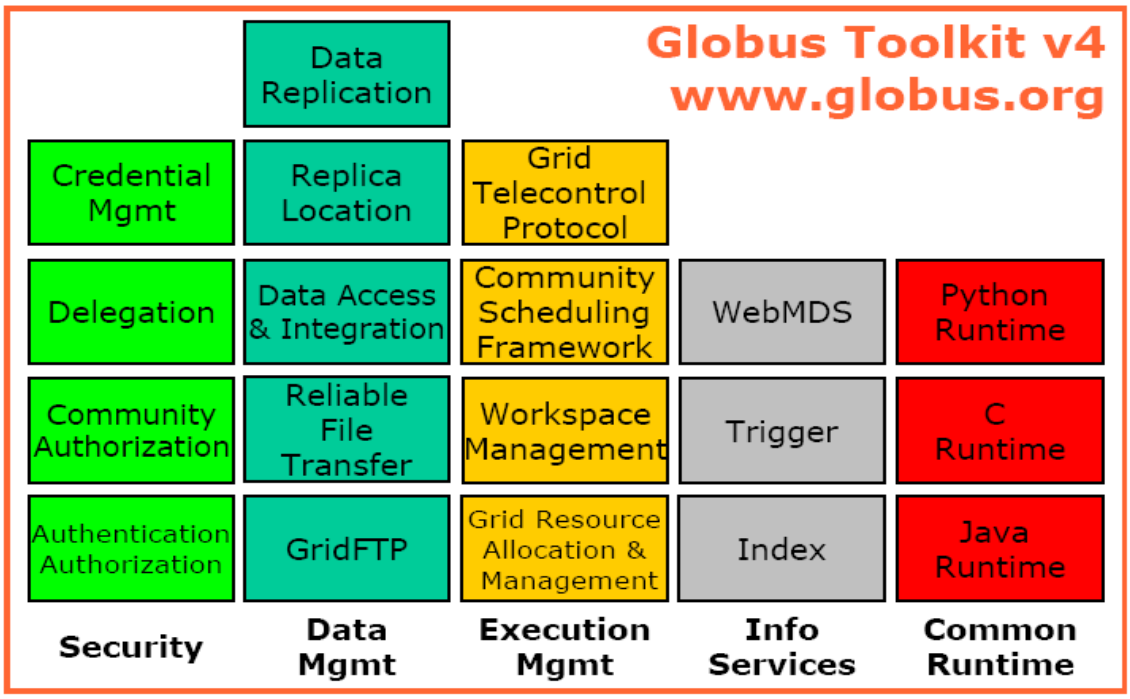

Figure 2.7 Globus Toolkit 4 services [2]

2.1.8 The Implementation of Xen Based Computational Grid

Xen-based computing grid consists of the following 3 clusters:

1 Mercury UVic Cluster.

2 Fate UVic Cluster.

3 NRC Cluster.

Babar jobs are built for SLAC and they are submitted through metascheduler either condor-submit or WS-GRAM submission. Metascheduler schedules the best resource and submits jobs to these clusters. Once jobs are finished, output is brought back to the submit machine using globus-url-copy.

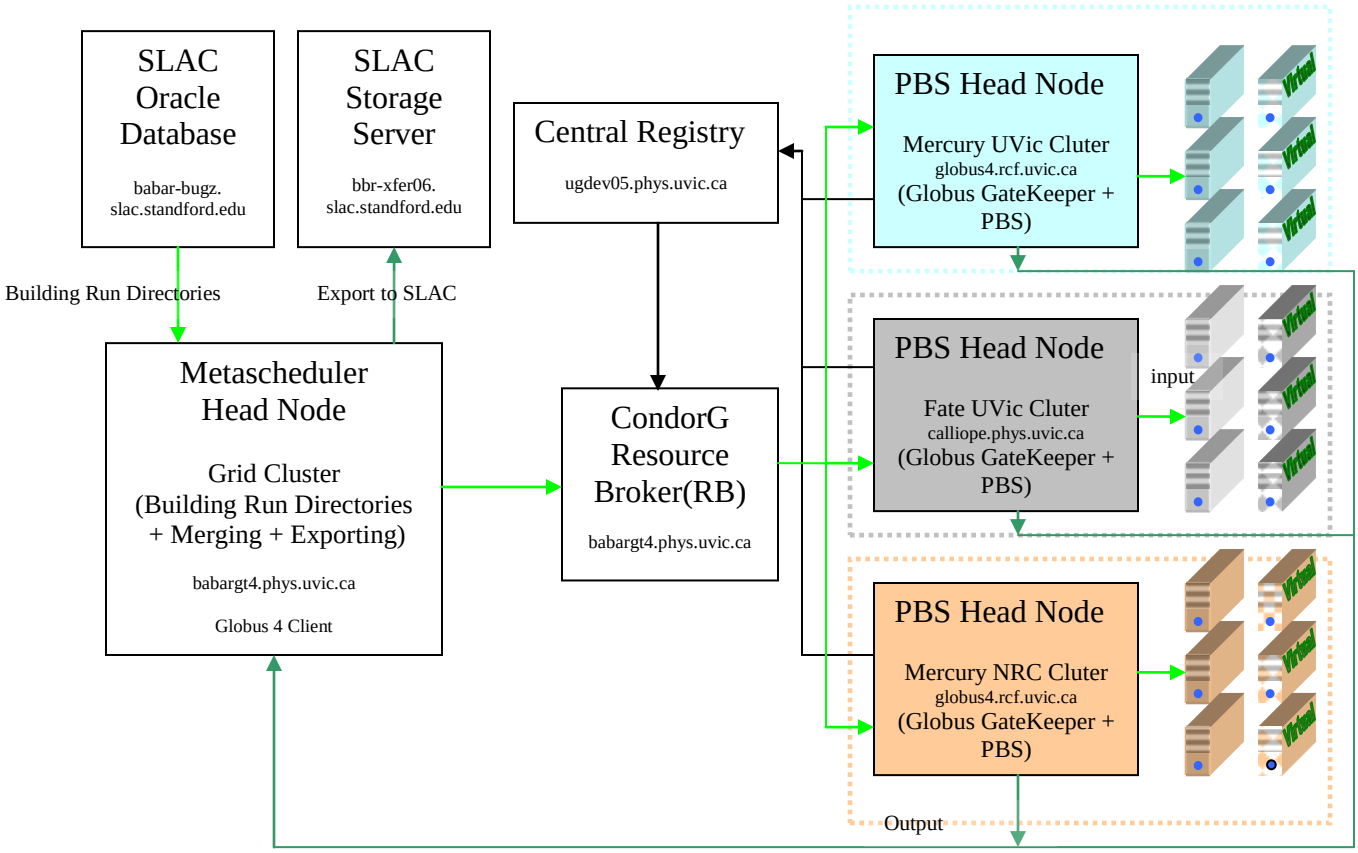

Figure 2.8 Xen-based Computational Grid Implementation [4]

2.2 Xen-based Worker Node Performance Test

Both network performances and CPU performance on the Xen based grid are compared with the corresponding tests on the native host domain.

2.2.1 Network Transfer Rate

The network performance is studied by measuring the transfer rates as a function of number of parallel streams and compared to the native Linux system as shown in Figure 2.9. The transfer rate was close to 25% slower than the native system at different number of parallelism. A slow down in network transfer is expected due to the following main reasons:

- The same network card is handling network traffic for both the domain0 and domainU, competition occured.

- The domainU traffic went through a software bridge which is an extra step comparing to the native domain.
- Operating System is running slower in the domainU since the system file system is running in image file.

Although a slow down in network transfer is expected, about 25% slow down is still a surprise. Further testings are carrying out to find the main factor of the slow down. The tranfer rate at number of parallelism of 5 shows comparable transfer rate, this suggests that the overhead on the guest operating system might be one of the main reasons of the guest network slow down issue.

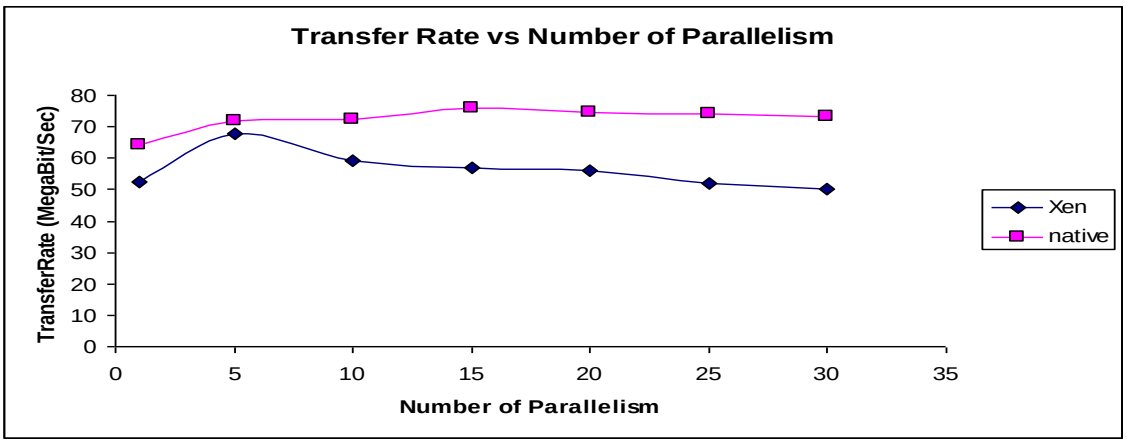

Figure 2.9 Network Transfer Rate Comparision

2.2.2 CPU Performance for Real Babar Job Validation Runs

The CPU performance was carried out by running real Babar HEP vlidataion jobs, and comparing their run time with the corresponding run time taken on the Linux native hosts. Such a comparison of run time is presented in Figure 2.10. Considering the fact that resources are split between domain0 and domainU, a 25% slow down of validation job run is not a surprise.

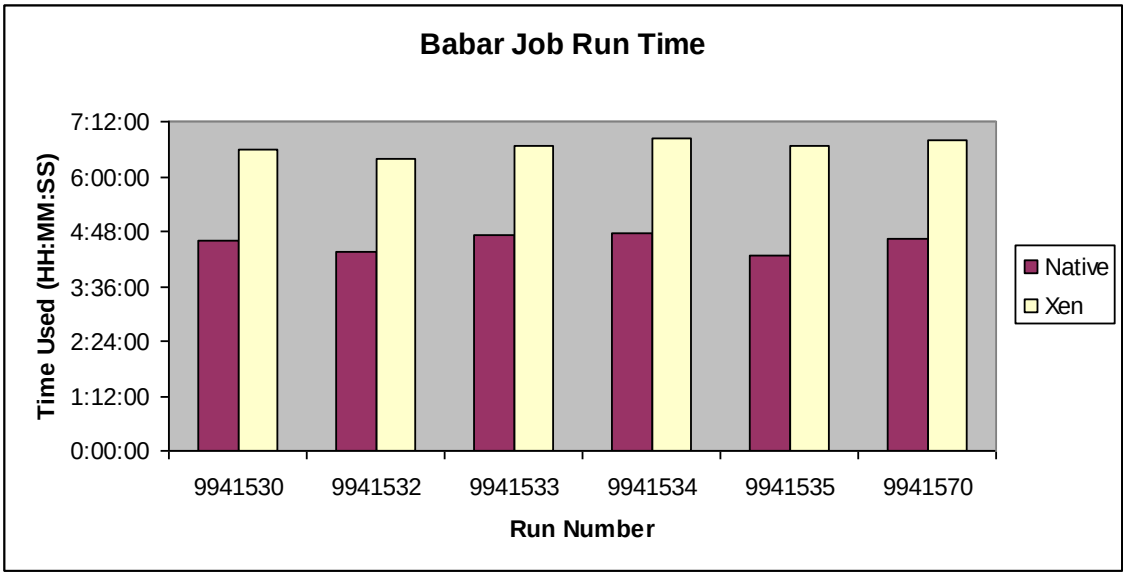

Figure 2.10 Babar Job Run Time Comparison

2.2 Condor Web Service SOAP Client

2.2.1 About condor

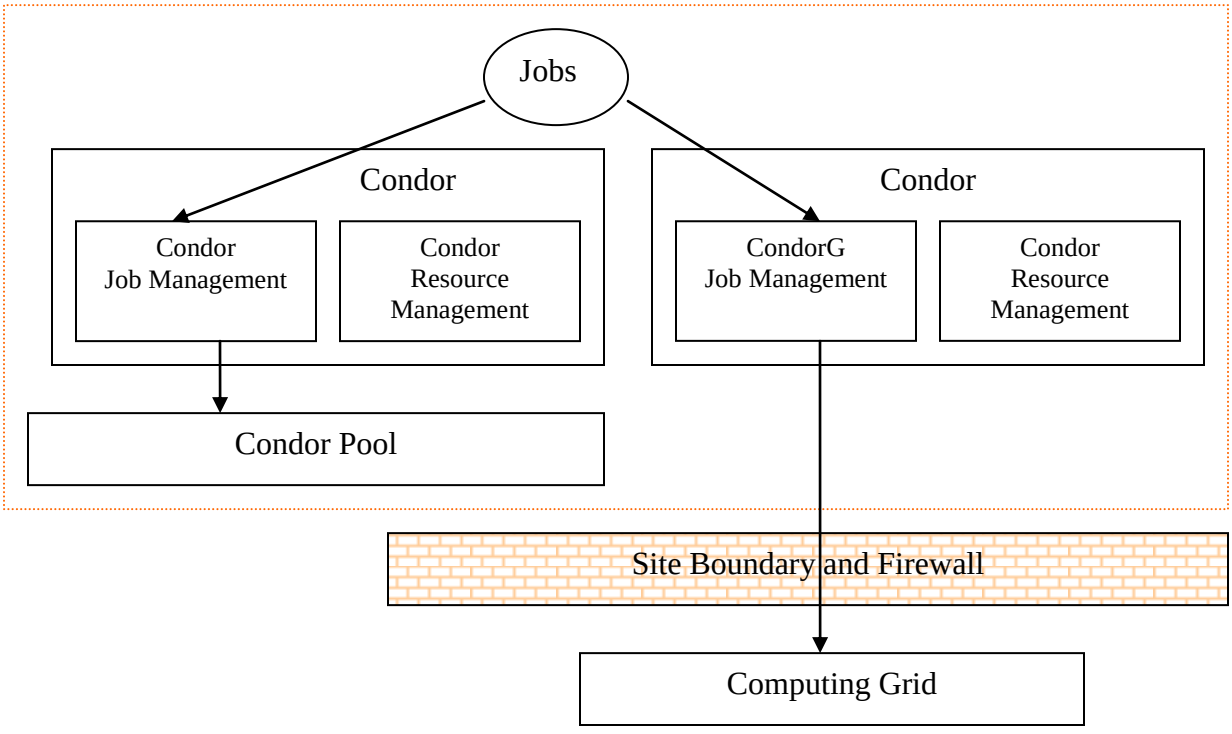

Figure 2.11 Condor Job Submission

2.2.2 About Web Service

Web Services are services that are hosted on a Web server. A client request the service through Web API, the execution is on the host machine, and result is then returned to the client. The supported operation is written in Web Services Description Language (WSDL) in server. A client can invoke the service based on the WSDL. The message is exchanged between client and server in XML format that follows the Simple Object Access Protocol (SOAP) Standard.
The advantage of Web Service over other remote procedure call is that it is platform independent, and the network traffic is http/https over TCP/IP, thus can easily pass through firewall. This makes it ideal for submitting jobs over the web or through site boundary to remote clusters.

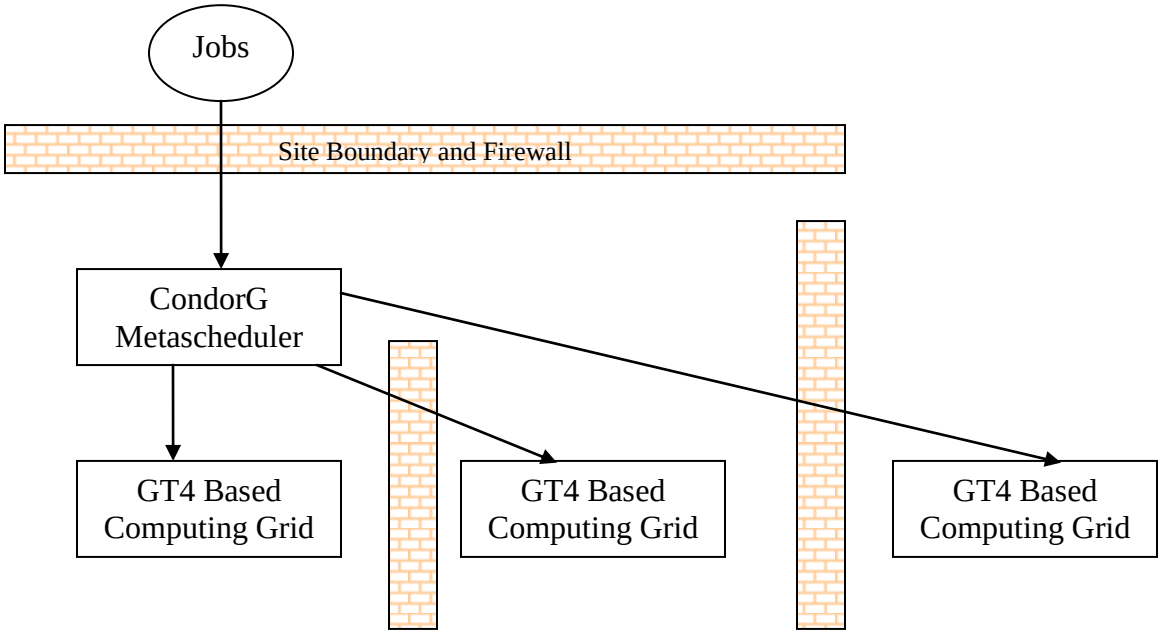

Figure 2.12 GT4 Job Submissions through Condor G

2.2.3 The work flow

The main job submission steps are shown in Figure 3.1. With the classes discussed in the 2.2.4, the application can be written quite easily.

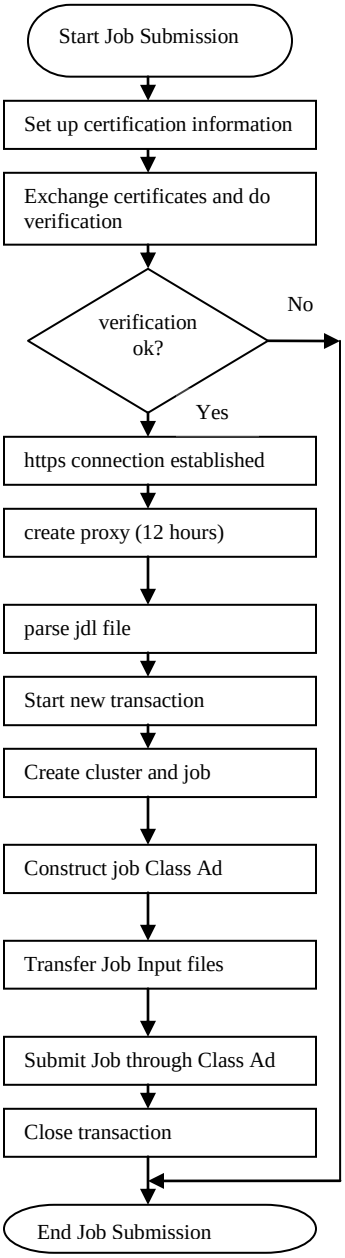

Figure 3.1 Job Submissions to Condor Pool Flow Chart

2.2.4 Class design

Condor Java Package [11]

Condor Java Package is generated from the WSDL files for the Collector daemon and Scheduler daemon by using the Apache axis tools. The Apache axis WSDL2Java tool can generate the Java client side stub through the WSDL files. The package generated consists of many java classes. It is difficult to use those many classes effectively to build client-side programmes, thus a wrapper Java package was written by the University of Wisconsin Condor Team.

This API is supposedly has exposed all a client needs to operate condor through command line. It does include all the functionalities such as submitting jobs, check job status, and control jobs. But all these are limited to Condor Pool only. Although it is claimed to be possible, submitting jobs to Grid is not directly supported.

Birdbath Java Package

Birdbath Java Package is the wrapper on Condor Package. It is aimed to simplify the client-side programme and it does make it much simpler. Since the Birdbath Package is still at the research stage, there is still no official release yet. It is expected to be included in the next release.

There are some limitations on this, to name a few:

1. Since Condor Java Package is generated from WSDL, the generated java code depends on the tool axis. With different version of axis, the java code is slightly different. If a user uses the java code directly, there is no issue. But in order to use the Birdbath package, the version of axis has to be the correct one 1.4 which is the recent major release.

2. The wrapping is not complete, that means it is not possible to do some of the jobs without access the Condor Package directly.
3. There is very little documentation on how to use it.

Application Classes

CondorJobSubmitter

This class responsible for all the works that related to submitting a job, such as Parsing JDL file, construct ClassAd, Stage-in input files etc.

CondorJobStatus

This class responsible for quarrying the server for job status, pausing and resuming jobs, cancelling jobs, and retrieving job output files.

2.2.5 Challenges

Authentication Issue

Authentication is very important for a Computing Grid. Condor supports many authentication methods such as File System (FS), Secure Sockets Layer (SSL), Grid Security Infrastructure (GSI), Kerberos, etc. But Condor SOAP API only supports SSL. Early versions of Condor is known to have problems with SSL, Recent versions are claimed to be okay. Testing on SSL through SOAP API succeed without issue.

JDL parsing Issue

Job Description Language (JDL) is used to submit jobs to Condor through command line. Through SOAP API, only ClassAd is accepted. The process of transferring JDL into ClassAd looks simple; it is not a straight forward process. A third party (glite) parser was used in this project, and in order to construct CalssAd successfully, knowledge on packages such as condor, classAd is also needed.

Submitting to GT4 based Grid issue

In order to submit jobs to GT4 based grid through CondorG, Proxy needs to be created and transfer to a local folder first. In the Command Line way, we issue Grid-proxy-init to create a proxy in /tmp folder. In SOAP, we do not have access to the /tmp folder, the only folder we have access is the spool folder. The proxy will be transferred into the spool folder, and the parameter of the proxy location and proxy file-name in ClassAd will be pointing to the file in the spool folder before the job is submitted.

Except for the proxy, there are other input-files will be also transferred into the spool folder before the job is submitted.

2.2.6 Project Status

Functionality Status:
Job submission to Condor Pool on ugdev01 with SSL authentication enabled has been tested successfully. All the job control functions also work without issue.
Job submission to Condor Pool on babargt4 with SSL authentication has been tested, but job submission to GT4 based Grid has not been tested. This is also one of the most critical parts.

Documentation Status:
Code has not been documented properly since it is still at the stage of testing functionalities.
User Interface is still on command-line with minimum Graphical User Interface.
Debug needs to be done.

2.2.7 Recommendation

Separate OS image and Application software image, transfer and mounted when needed.
Advantages:

1. OS image can be kept small and standard.
2. No need to enlarge the OS image (a risky process)

3. Extra space can be added in when needed.
4. Some application can be “installed” in separate image and “mounted” at run time.

Disadvantages:

1. Possible performance hit since OS and other spaces are in different image.
2. Some of the software can not be mounted this way.

Two ways to mount extra images

1. During guest creation time
2. Mount on the fly

2.3 Conclusion

The creating of Xen based Grid and evaluation of the performance has been done and premium result data was collected. We are confident to say that it is feasible to create Xen-base grid with acceptable performance to run HEP applications.

The Condor SOAP API Client has not been completed, but it is very close to success. It is worth to spend a little bit extra effort to complete the project.

2.4 Future Work

2.4.1 Condor Web Service Project:

- Complete the functionality test of submitting jobs to GT4 based Grid

- Work with Andre to incorporate the SOAP into Job Submission Client. Andre is working on this too.

2.4.2 Xen-Based Virtual Worker Node

- Set up Xen-based Virtual Worker Node on Mercury Cluster and cluster in NRC.
- Run Babar application on Virtual Worker Nodes in the above mentioned Clusters.
- Run Babar application on Virtual Nodes on dedicated Disk to evaluate how much the disk affects the performance.
- Compare and analyse the performance in all locations.

2.4.3 New Things that are related:

- Putting software (such as PBS, Globus Toolkit, and Babar Application) in image-file and mount it on guest (instead of installing statically or mounting from network). Finding a general way of "installing" software on virtual machines. In this way, the preparation of OS and Application can be separated, thus flexible.
- DRMAA API (Binding in Java and C) is an alternative to Condor SOAP API. It is the future standard way of submitting and control jobs to one or more Distributed Resource Management Systems. Condor has also implemented this API. There is sample code (in Java) for download from sourceforge.com.

3. The List of References

[1] Paul Barham, Boris Dragovic, etc. Xen and the Art of Virtualization

[2] Globus Toolkit 4 User Manual www.**globus**.org/**toolkit**/docs/**4**.0/

[3] Condor User Manual http://www.cs.wisc.edu/condor/manual/v6.7/

[4] Ashok Agarwal, Ron Desmarais, etc. Babar MC Production on the Canadian Grid using a Web Services Approach

[5] Ferry Fleury myproxy http://grid.ncsa.uiuc.edu/myproxy/sessions/SSOUtils.java

[6] Xen User Manual http://xen.xensource.com/files/xen_user_manual.pdf

[7] Xen Networking http://wiki.xensource.com/xenwiki/XenNetworking

[8] Angela Norton Installing Xen https://wiki.gridx1.ca/twiki/bin/view/Main/InstallingXen

[9] PBS Administrator Guide www.mta.ca/torch/pdf/**pbs**pro54/**pbs**proag_5_4_0.pdf

[10] Building a Beowulf Cluster: OpenPBS Overview and Configuration
http://www.ats.ucla.edu/rct/clustering/software/job_management_systems/pbs_overview_and_configuration.htm

[11] Jeff Mausolf Manage grid resources with Condor Web services
www.ibm.com/developerworks/edu/gr-dw-gr-ws**condor**.html

Glossary

Cluster	A group of connected computers work together as if there is one computer
Condor	Software developed by University of Wisconsin to do distributed computing
Extensible Markup Language	User defined description language
Globus Toolkit	software to provide grid infrastructure
Globus Toolkit 4	version 4 with Web Service capability
Grid	Consists of many clusters, accept computing jobs from user. Acting like a supercomputer.
High Energy Physics	for example, Particle Physics, Nuclear Physics
Job	Computing programme needs to be excuted.
Metascheduler	A Grid scheduler that manages many distributed computing resources and jobs
Portable Batch System	software used to form local cluster
Proxy	A file contains a time-limited certificate and passwordless private key
Registry	A centralized store. Information such as resource type, resource status are saved.
SOAP	XML based protocol used to invoke web services
SSL	Secure Socket Layer
WSDL	Web Services Description Language