University of Victoria
Faculty of Engineering
Spring 2003 Work Term Report

# Running the ATLAS Simulation Across the Grid Canada Testbed

Department of Physics
University of Victoria
Victoria, British Columbia

Manj Benning
9804096
Electrical Engineering
mbenning@engr.uvic.ca

April 25, 2003

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)

| Signature | Position | Date |
|---|---|---|
| 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.

#1537 Bay Street
Victoria, British Columbia
V8N 2B3

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

April 25, 2003

Dear Coordinator,

Please accept the accompanying work term report entitled "Running the ATLAS Simulation Across Grid Canada".

This report is the result of work completed at the Department of Physics at the University of Victoria. During this, my fifth work term in the Electrical Engineering program, and first with the UVic Physics Dept., my assigned task was to continue the development of the Canadian Grid Computing testbed, and to determine the best way to run the ATLAS simulation across it.

Through the course of the term, I was given the opportunity to learn quite a bit about Grid Computing. The skills I have learned will undoubtedly assist me in future endeavors.

I have many people to thank for their assistance during the last four months, namely Dr. Randy Sobie, Dr. Bob Kowalewski, Dr. Ashok Agarwal, Jenny Allan, Dan Vanderster, Laura Kormos, Tayfun Ince, Ian Nugent, and Dominique Fortin.

Sincerely,

Manj Benning

# Contents

# List of Figures



# List of Tables

# Executive Summary

In the early 90's particle physics was the prime motivation for the development of the World Wide Web (WWW). Since then the WWW has become an invaluable resource for information and communication, used by millions of people all over the globe. Today, particle physics is again pushing the computing envelope, unifying computers, storage facilities, disks, and processing farms into large distributed computing grids built on the existing infrastructure of the internet.

The ATLAS project [1], to be completed in 2006, is the next major physics experiment. The unconventionally large amount of data and analysis that this experiment will require is forcing universities and other research institutes around the world to share their computing resources to overcome the ATLAS data challenge.

Grid Canada, together the University of Victoria's particle physics group, have brought various computing resources around the country together to act as a single super computer, capable of running applications such as the ATLAS simulation.

The ATLAS simulation will be used to aid in the design of the ATLAS detector and was run across the Grid Canada testbed using various methods. Each of these methods were tested to discover which one yielded the fastest total runtimes and highest CPU efficiencies. Throughout these tests, the UVic grid group learned about the limitations and difficulties involved in running large CPU intensive applications over a distributed grid environment.

[1] Stands for A Torodial LHC Apparatus

# 1 Introduction

Science today is pushing computing power to its limits. Traditional computing methods cannot keep up with the shear magnitude of data and CPU power demanded by modern day experiments. The next generation particle physics experiment, Large Hadron Collider (LHC) at CERN in Geneva[2], is expected to produce a petabyte of data per year [1]. About 1000 times the data that CERN can handle[3]. The analysis and storage of such a large amount of data is a daunting task that any single computing site cannot handle. A solution to this problem is to utilize CPU processing power and storage facilities around the world on a unified distributed grid.

A distributed computing grid consists of high performance computers, processor farms, disks, databases, informatic systems, and collaborative tools all linked by a high speed network [2]. Created on the "Why reinvent the wheel?" philosophy, computing grids are a relatively inexpensive way to perform a large computing task which would swamp any single computer bank. Computing grids are using existing infrastructure around the globe to create a new generation of super machines. Similar to the WWW, users of the grid may exploit grid resources unaware and unaffected by their geographical location. Grids are being used by various collaborations, such as particle physics, across many countries.

The Canadian grid effort, Grid Canada, consists of several Canadian Universities, one U.S College and several research institutes across Canada. Currently there are a total of twenty two computers at eleven sites which are participating in Grid Canada. Three other sites are willing to contribute but are not quite ready. The grid group at the University of Victoria (UVic) is leading the development of the Grid Canada testbed.

The Canadian grid effort's primary motivation is particle physics simulation. Researchers at UVic are involved in the ATLAS project, a high energy particle physics collaboration of 1800 scientists and engineers at 150 universities in 34 countries. The ATLAS project hopes to uncover the fundamental nature of matter[1]. The grid group at UVic have been running a simulation of ATLAS over the Canadian grid to develop expertise in grid technology, learning how to manage a grid, and learning how to optimize particle physics applications in a grid environment[2].

The Atlas simulation was run over the Grid Canada testbed using different methods. The purpose of this report is to: Describe the ways in which the simulation was run, analytically compare the different methods, and present the best method with proof based on experimentation. This report will conclude some limitations of the Canadian grid and offer some insight into which direction Grid Canada should head.

# 2 Discussion

Before discussing the different methods for running the ATLAS simulation, some background information on Grid Canada's infrastructure will be explained.

[2]http://www.cern.ch

## 2.1 Grid Canada Achitecture

The major components of the grid involved in running an Atlas job are: The Globus Toolkit, AFS, the local machine or submission machine, the remote machine or execution machine, and the network.

### 2.1.1 The Globus Toolkit

Pioneered by Ian Foster and other developers at the Globus Project, the Globus toolkit or Globus is an open-architecture standard for grid development. Currently, Globus is widely accepted as the standard 'bag of services' for building computational grids and grid-enabled applications[5]. The Globus toolkit provides a variety of low-level services which are used to grid-enable resources across Grid Canada. The three main Globus sevices are: Grid Resource Access and Management (GRAM), Metacomputing Directory Service (MDS), and GridFTP (Grid File Transfer Protocol).

The GRAM service is comprised of the globus gatekeeper and the globus job manager. The gatekeeper is responsible for authenticating a user to the remote resource. Before a job is submitted, the user of the grid must acquire a grid proxy using the *grid-proxy-init* command. Each users proxy exists for a default time of 12 hours and can be used to access any permitted grid-enabled resource. When a globus client submits a job to a remote machine or globus server, the gatekeeper checks the clients proxy against a gridmap file that exists on the server side. The gridmap file is a list of permitted users and associated directories to which each user has access to. Once the user is authenticated, the job manager executes the job and sends back the result to the client machine.

The MDS service provides an infrastructure to organize and store dynamic and static information about hardware components on the grid, such as the network, data storage facilities, and computers. Each grid-enabled resource communicates hardware information such as CPU load, RAM, and hard disk space to the Grid Resource Information Service (GRIS) on each globus server. The GRIS service on each machine in turn reports to a Grid Index Information Server (GIIS). Grid Canada runs a GIIS server at giis.gridcanada.ca; information of all machines registered with Grid Canada is available here[6].

The GridFTP service is used for transferring data across the grid. Similar to standard File Transfer Protocol (FTP), GridFTP supports file transfer over wide area networks with basic security. In addition to basic FTP, GridFTP has support for reliable transfer of very large data sets. The number of parallel streams may be specified, greatly increasing the transfer time. The ATLAS job submissions use up to 24 parallel streams to transfer the input and output data. Stream numbers in this range can increase transfer speeds upto eight times more than single stream transfers. Furthermore, GridFTP supports http, simple file transer, and third-party transfers. Third party transfers allow a grid user to specify a client to server transfer while at a third uninvolved machine. Third party transfers can be useful if a grid web portal is being used to submit jobs and the data is not available locally on the web server machine.

### 2.1.2 AFS

AFS stands for the Andrew File System. It is a distributed file system available over both local and wide area networks. Via its complex authentication mechanism, permitted users can log on to an AFS server and access remote resources. AFS is similar to NFS, Network File System, but can also be used over a wide area network. Each Grid Canada resource runs the OpenAFS client so they can exploit AFS trees over the grid.

To use the file system a user must issue the *klog* command, identify themselves and specify which cell they want to access. The cell is the name of the AFS resource; UVic maintains the phys.uvic.ca cell and Stanford Linear Accelerator maintains the slac.stanford.edu cell. Each permitted user of an AFS cell has an account on the AFS server and has specific permission set for each directory on the tree. Access rights can be tailored specifically for each user.

For AFS access over the grid, OpenAFS provides a means of authentication via a grid user's globus proxy. Instead of the *klog* command a *gssklog* command is issued. Through *gssklog* a user is not required to have an account on the AFS tree. Only a valid grid-proxy is needed. The gssklog daemon, which runs on the AFS server, effectively grid enables an AFS resource.

AFS is an integral part of running the Atlas simulation over the grid. During job execution a remote machine will always access the Atlas simulation software from AFS; this includes the 355 MB executable and the software libraries. In some cases the input data and output data is also accessed from AFS. To increase the job efficiency a second 'clone' AFS server was recently added. This distributes the load from the remote machine over two AFS computers instead of one. Tests are currently underway to determine whether the clone is actually increasing efficiency.

### 2.1.3 Local and Remote Machines

All ATLAS jobs are sent from a local machine, also known as the submission or client machine, to a remote machine, also known as the execution or server machine. All client machines need to have the globus client software installed inorder to communicate with grid enabled resources. Most of the computers using the grid have globus client 2.2.4 installed. Permission to use each grid resource also requires a valid globus proxy. To obtain a proxy the user must have a user certificate issues by the Grid Canada Certificate Authority[3]. The submission machine for every test performed within the scope of this report was chimera.phys.uvic.ca located at the UVic physics department.

Remote machines are the grid resources, used for their storage and idle CPU cycles. Almost all of Grid Canada's resource machines have globus 2.2.4 server installed on them. To run the ATLAS simulation all remote machines must also have the OpenAFS client installed inorder to access the AFS tree. Chip speeds for machines across the grid range from 450MHZ to 2GHZ and all machines that run the ATLAS simulation must have at least 355MB of RAM. This stipulation ensures that the ATLAS executable, of size 355MB, will be stored in RAM during execution and not swap memory. Simulations using swap memory tend to take a very long time to finish due to the high number of slow hard disk accesses.

[3]certificates are obtained from ca@globus.org

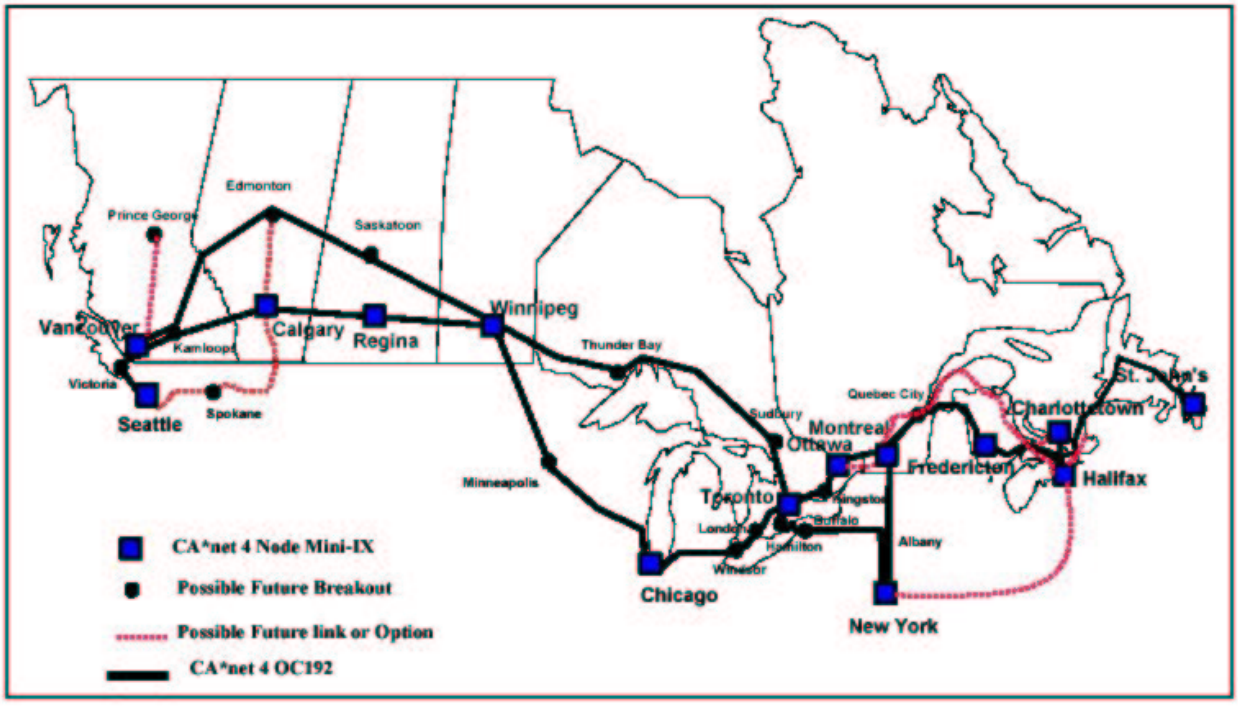


Figure 1: A map of the the Canadian fibre optical network backbone, CA*net 4

### 2.1.4 The Grid Canada Network

Canada's advanced network infrastructure is a solid platform on which to build the Grid Canada testbed. The work horse of Canada's highspeed network is CA*net 4, see figure 1. As one of the worlds fastest, the CA*net 4 fibre optical backbone spans from Victoria, BC to St.Johns, Newfoundland, allowing speedy transmission of large data sets pertinent to a grid environment. Capable of supporting upto 40Gbits/s, CA*net 4 connects research institutes which are on province specific regional grids[7].

UVic recently increased its connectivity to 1Gbit/s since it was connected via fibre to the Victoria Transit Exchange (VTE). The VTE is a meeting point at which institutions can buy a connection directly to the Vancouver regional network or CA*net 4 with 2.5Gbit/s connectivity. Effectively UVic physics could request a routing to CA*net 4 through only one hop at the VTE. However, more important political issues must be addressed before this connection would become reality.

## 2.2 ATLAS and the ATLAS Simulation

ATLAS stands for A Toroidal LHC Apparatus and LHC stands for Large Hadron Collidor and will be one of the particle detectors integrated in LHC. ATLAS will be responsible for observing the collision paths and communicating the information from the 14TeV, proton-proton collisions that LHC will perform. The ATLAS engineers and scientists plan for the ATLAS particle detector to built for 2006. Before then, much research, design, and testing needs to be done.

One major aspect of the design stage is to simulate how the detector will act before it is fully built. Physicists have developed software to perform these simulations. The full simulation will consist of several parts or 'data challenges', each requiring large amounts of storage and hundreds of CPU hours to run. Currently, only the first part of the ATLAS data challenge is being tested.

The first part is responsible for generating the background data or noise that will be prevalent in the actual ATLAS detector. Even though this is only the first part of the simulation it requires a large amount of data and is quite CPU intensive. This first stage of the ATLAS simulation requires a 355 MB executable, 2.5GB of input data and produces 2.1GB of output data. A simulation can take from 44 hours to 28 minutes depending on how it is run.

Because of its size and intensive CPU usage, the ATLAS simulation is a good candidate for testing Grid Canada's ability as a distributed computing grid. The ATLAS simulation was run over the Grid Canada testbed using three different methods: The Full AFS Method, The Partial AFS Method, and the Full GridFTP Method.

## 2.3 Running the ATLAS Simulation

The developers did not design the ATLAS simulation software to run efficiently over a computing grid. This has posed a tough challenge to the grid computing group at UVic, who has had to figure out the fastest and most efficient way for remote machines to run multiple simulations while reading the input data and writing the output data back to UVic. Three methods were used to run the simulation. The following section outlines each method. For a more indepth explaination on how each simulation is run and the different scripts involved see appendix A.

### 2.3.1 The Full AFS Method

This method requires that both the 2.5GB of input data and the simulation software be read off the AFS tree located at the physics department at UVic. The ATLAS executable, *atlsim.exe*, along with the script responsible for running the executable, *dc1.pileup.standard.v2*, are copied over to the remote resource via a basic single stream linux copy command. A dynamic link is then set up to read the software libraries from the AFS tree during simulation.

As the simulation runs, output is continuously written to an AFS cache on the remote machine. The remote cache sizes are typically in the range of 80MB. When the cache fills, the remote machine copies the contents of the cache to the AFS tree via a basic single stream copy command and erases the remote cache. Once completed, the simulation leaves the output on AFS at UVic.

### 2.3.2 The Partial AFS Method

Different to the Full AFS method, the Partial AFS Method requires the input data, as a whole set, to exist on the remote machine during execution. The ten input data files get sent over to the remote resource as a single tarred and compressed 1.27GB file using 24 stream gridFTP. Before the simulation begins, the remote machine untars and uncompresses

the input data, copys over the ATLAS executable and the *dc1.pile.standard.v2* script, and creates the dynamic software link to AFS.

Similar to the Full AFS method, the output data is cached on the remote machine and dumped out to the AFS tree periodically. The output is left on the AFS when the simulation completes.

### 2.3.3 The Full GridFTP Method

The Full GridFTP method takes many small steps to run an ATLAS simulation. Similar to the Partial AFS method, it requires the complete input data set to be on the remote machine during execution. Similar to both previous methods, the software, including the executable, the ATLAS script and the dynamic link, are conceived from AFS. However, unlike the previous two methods, the output data is written directly to the remote machine's hard disk; Then it is sent back to UVic as one file. Here are the steps involved in the Full GridFTP method:
1. Tarred up and compressed input data is sent over to the execution (remote) machine from the local machine via 24 stream gridFTP.
2. Input data is untarred, uncompressed on remote machine.
3. The simulation is run (executable and ATLAS script copyed over from AFS).
4. Output data is tarred, compressed and saved in a different folder from where the output was written. The original output is removed from the remote machine.
5. The tarred and compressed output is sent back to the local machine via 24 stream gridFTP.
6. Output is untarred and uncompressed on the local machine.
AFS is only used for the software in this method and the remote AFS cache is not used. For more explaination on how the different simulations are run, see appendix A.

## 2.4 Testing the Different ATLAS Simulation Methods

The two key parameters in determining the merit of a simulation method are the total time it takes to run the simulation and the CPU efficiency. From the moment the job is submitted from the local machine to when the output is fully available on either UVic Physics AFS or the local machine is the total time of simulation. The total time can be determined by preceding the submission statement with the UNIX *time* command.

The output of the *time* command displays the real, user, and system times. The real time is the total amount of time spent executing just the ATLAS simulation, including I/O operations; This is step 3 in the Full GridFTP method, the whole time in the Full AFS method, and execution time not including the input data transfer and untarring for the Partial AFS method. The user time is how long the remote CPU spent computing specifically the ATLAS machine code, this does not include any I/O operations. The amount of time spent on performing system tasks is the system time. On a local 1GHZ Pentium III machine the simulation took 1 hour and 41 minutes. This local result will be used to compare the runtimes of the various methods over the grid.

CPU efficiency is determined by preceding the ATLAS dc1.pileup.standard.v2 wrapper script call with the *time* command. More information on the ATLAS wrapper scripts can

| Site | Resource Name | Successful / Total Num. Jobs |
|---|---|---|
| URegina | sisyphus.phys.uregina.ca | 1 / 1 |
| URegina | lafite.phys.uregina.ca | 1 / 1 |
| URegina | lambro.phys.uregina.ca | 1 / 1 |
| NRC IIT | ontario.iit.nrc.ca | 0 / 1 |
| NRC IIT | superior.iit.nrc.ca | 1 / 1 |
| NRC IIT | huron.iit.nrc.ca | 1 / 1 |
| UBC | axen.physics.ubc.ca | 0 / 1 |
| Carleton | obgp2.physics.carleton.ca | 0 / 1 |
| Carleton | obgp3.physics.carleton.ca | 0 / 1 |
| BC Ministry of M. S. | gridtest.net.gov.bc.ca | 1 / 1 |
| PFC | tree.pfc.forestry.ca | 1 / 1 |
| UVic | grid.phys.uvic.ca/condor | 5 / 5 |

Table 1: Machines used for the Full AFS tests

be found in appendix A. CPU efficiency can be written as:

$$E = (T_{user} + T_{system})/T_{real} \tag{1}$$

To determine the fastest and most efficient method, a series of tests were performed. Locally, the ATLAS simulation acheived an effciency of 99%. This optimal value will be used to compare against running ATLAS over the grid.

### 2.4.1 Testing the Full AFS Method

Testing of the Full AFS method was performed in the previous workterm by Dan Vanderster. This section will summarize his results and bring it into context with the other two methods that were implemented during this workterm. Much more experimentation was not performed with the Full AFS method due to its poor performance.

On December 6, 2002 16 jobs were submitted to the Grid Canada testbed at a variety of sites. Table 1 shows a full list of the sites and the number of simulations. Four of the jobs did not complete mostly due to network complications. Via the Full AFS method, the ATLAS simulation ran suprisingly slow. At the UVic grid cluster, where there is no latency between the resource and AFS, the total time for execution was about 21 hours. Figure 2, graph number two shows the 'Total Runtime' in seconds versus the 'Latency' in milliseconds. It is apparant that as latency is introduced between the remote resource and AFS, the total runtimes tend to increase.

Particularly disappointing are the CPU efficiencies for the Full AFS method. The greatest CPU effcencies, experienced at the UVic grid nodes, were 10%. According to figure 2, the CPU efficiencies drop off almost exponentially as the latency from the remote resource to AFS increases.

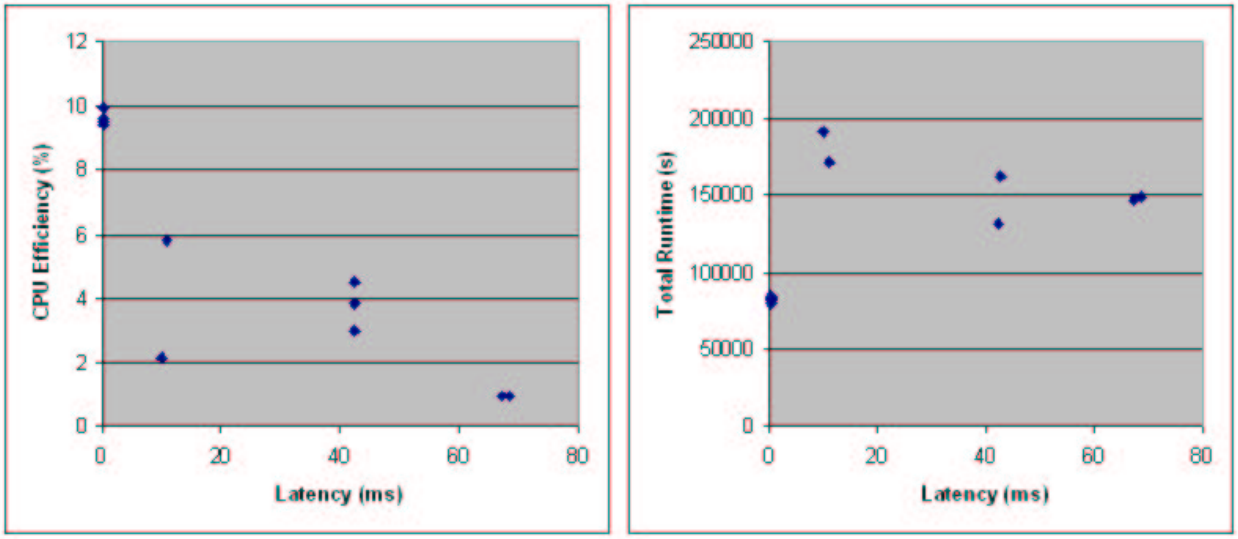


Figure 2: CPU efficiency (left) decreased and Total Runtime (right) increased as latency to executing resources increased. Additionally, the peak CPU efficiency was only 10%, which is one tenth of that achieved when running the ATLAS application locally.

### 2.4.2 Discussion of Results

Running the ATLAS simulation with the input and output data on AFS is simple and convienient. However, the total runtimes are too high and the CPU effciencies are too low. Unlike the Full GridFTP method, the Full AFS method requires minimal scripted commands to run an ATLAS job. Since data transfer to AFS is automatic, the local or submission machine has little involvement except to just send over an execution script to the remote resource. The ease of use of this method is overshadowed by its lengthy runtime.

The Full AFS method's unacceptable long runtime is due to the latency of reading and writing of data over a network. A significant portion of the total runtime can be attributed to the remote resource making contact to the AFS server over the network while reading the 2.5GB of input data. As the application makes more connections to AFS the single ping latency, in the millisecond range, adds up and contributes to the overall runtime. The low CPU efficiencies can be explained by the considerable amount of time the CPU spends waiting for I/O operations. An increase in I/O operations increases the real execution time thus decreasing the efficiency.

The other significant contribution to runtime is the writing of output to AFS. As a remote job runs, the output is written to the local AFS cache on the remote machine. When the cache fills, 80 megabytes is transferred to AFS. Figure 3 shows the incoming and outgoing traffic streams for the UVic AFS server over a 24 hour period for a Full AFS method simulation to a remote resource. Outgoing data rates, or the reading of input data, slows down as data is written to AFS. On average, the remote job has to slow down almost to a halt every half hour to accomodate for the outflow of output data[2].

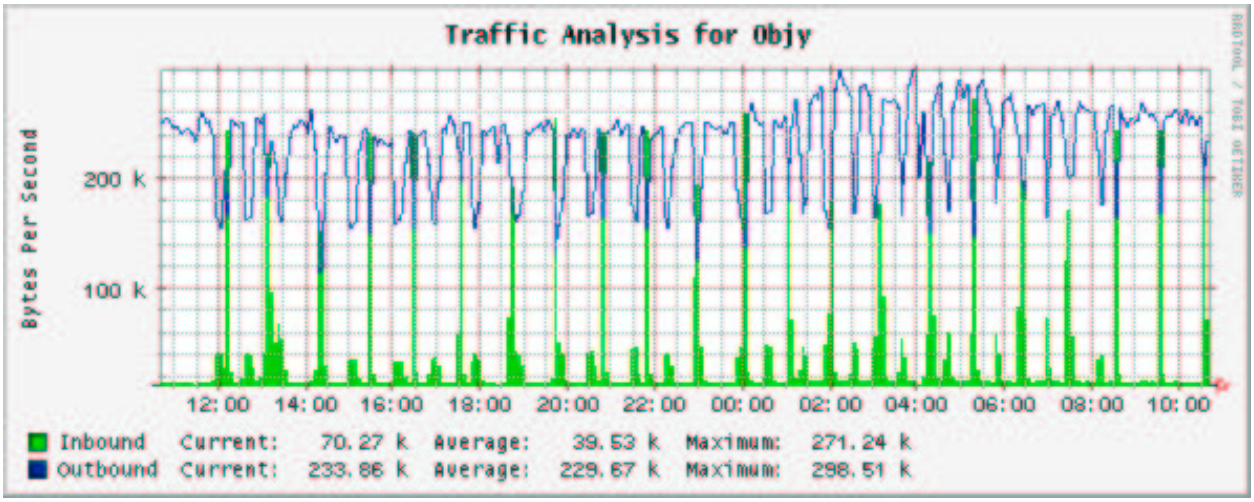


Figure 3: As the AFS cache gets dumped to AFS the outgoing data rate to the remote machine slows down

| Method | Number of Simultaneous Jobs | Total Runtime | CPU Efficiency |
|---|---|---|---|
| Partial AFS | One | 1hr 13min 53sec | 21% |
| Full GridFTP | One | 46min 18sec | 92% |
| Partial AFS | Three | 4hr 55min 2sec | 16% |
| Full GridFTP | Three | 1hr 43min 53sec | 81% |

Table 2: Total runtime and CPU efficiency were observed for the Partial AFS and Full GridFTP methods with one and three jobs running at the same time

### 2.4.3 Testing the Partial AFS and Full GridFTP Methods

The merit of the Partial AFS and Full GridFTP methods were determined using similar testing procedures unlike the tests performed on the Full AFS method. Each method was run once to completion on superior.iit.nrc.ca, a 2GHZ Pentium IV, at the National Reseach Council (NRC) in Ottawa, Canada. The real, user, system, and total execution time was observed aswell as the CPU efficiency. Then each method was submitted to superior.iit.nrc.ca again aswell as two other remote resources simultaneously. The real, user, system, and total execution time and CPU efficiency were again observed for the superior.iit.nrc.ca job. The two methods were directly compared inorder to conclude which one yielded the smaller total runtime and higher CPU effciency.

Table 2 displays the total runtime and CPU efficiency of each method with one and three jobs running simultaneously. The Partial AFS job ran in about 1 hour and 13 minutes while the Full GridFTP job took about 46 minutes to finish. The GridFTP method for a single job ran faster. Furthermore, running a single job via the Full GridFTP method yielded an efficiency more than four times that of the Partial AFS method.

The results from the triple submission tests confirm that the Full GridFTP method is the best solution to running ATLAS over a grid. Running three simulations at once with the Partial AFS method increased the total runtime four fold to running a single

job. The efficiency was also compromised down to 16% from 21%. On the other hand, running three ATLAS jobs with the Full GridFTP method only increased the runtime of the superior.iit.nrc.ca job by about one hour or two fold and dropped the CPU efficiency from 92% to 81%.

### 2.4.4 Discussion of Results: Partial AFS Method

Running the ATLAS simulation via the Partial AFS Method is almost as simple as running it via the Full AFS method and produced significantly better results. A level of complexity is added to the submission scripts due to the sending and untarring of the input data on the remote machine. However the output data is conveniently written to AFS.

Compared to the ATLAS runtime of 101 minutes on a local 1GHZ machine at UVic, the 73 minute runtime of the single Partial AFS job, on the 2GHZ machine at NRC, scales up quite well. However, there is still a significant amount of time spent contacting AFS to write the output data. Figure 4 shows the AFS traffic characteristics for a Partial AFS job. The output is being written to AFS almost consistently over the course of the simulation. Although not proven, it is a possibility that the remote job slows down when the AFS cache, on the remote machine, is dumping out. Since the AFS cache is dumping and filling up constantly throughout the job, the entire simulation runs slower than the Full GridFTP method.

The Partial AFS method showed weakness when mutiple jobs were submitted. It took almost 5 hours to run a simulation at superior.iit.nrc.ca when three other remote jobs were running across the grid. It is obvious that the traffic bottleneck at AFS is the cause. The right side of Figure 5 graphs the AFS traffic throughout the triple job submission. A bottleneck is apparant as each remote machine competes for the AFS server. If a remote resource has filled its local AFS cache and the AFS server bottleneck is preventing it from emptying, there is a strong possibility that the ATLAS simulation will come to a halt until the remote machine has space in the cache to write more output. Interestingly, the efficiency only dropped by 5% when three jobs were run at once. It is fair to infer that a greater number of simulations running simultanously will lead to longer total runtimes due to the bottleneck at the AFS server.

### 2.4.5 Discussion of Results: Full GridFTP Method

By far the most complex method, running an ATLAS simulation via the Full GridFTP method is speedy and the most CPU efficient. Referring to section 2.3.3 six steps are involved in running the ATLAS simulation and retrieving the output data. In comparison, the Full AFS method requires a single step to submit the job and the Partial AFS method requires two. However, the Full GridFTP is by far the most efficient and fastest.

A single job run on superior.iit.nrc.ca, a 2GHZ Pentium IV in Ottawa took 46 minutes and 19 seconds at an efficiency of 92%. Scaling that down to a 1GHZ machine you get a runtime of 92 minutes and 40 seconds. Astoundingly, this time is still faster than the 101 minute total runtime for the local UVic machine! If superior.iit.nrc.ca was running at 1GHZ with a Pentium III architecture, the remote runtime may be longer than the local runtime. However, this is by far the most promising result for running the ATLAS simulation.

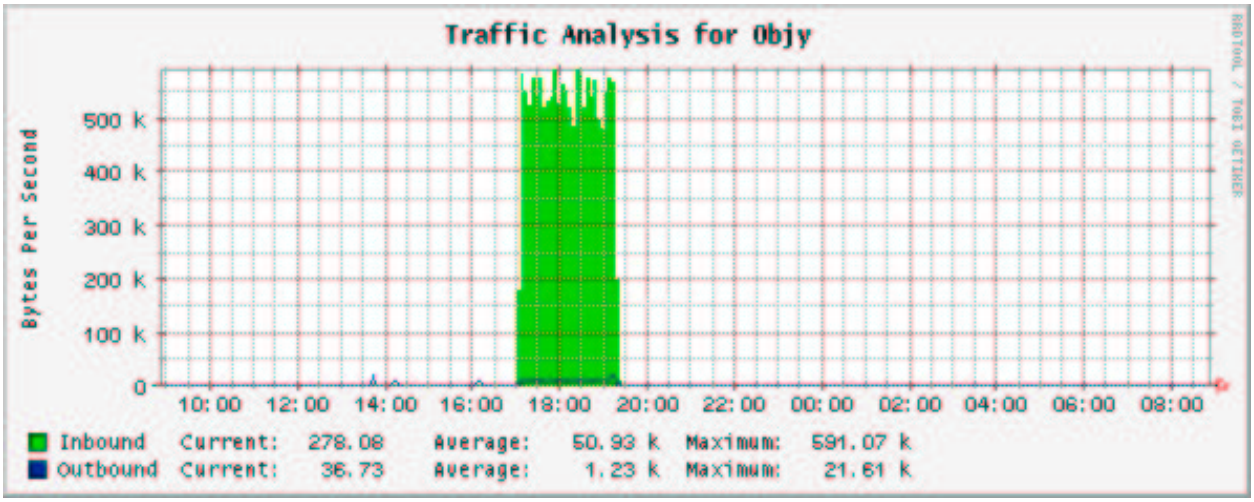


Figure 4: It is a possibility that dumping the output to AFS slows down the job execution on the remote machine

Running three jobs simultanously via the Full GridFTP method increased the total runtime by a little more than two fold and decreased the efficiency by 11%. In comparison the Full GridFTP method dominated over the Partial AFS triple total runtime of almost 5 hours. The large 11% drop in efficiency is an abnormal result and can be attributed to network inconsistency. Further tests are discussed in this report to discover the true characteristics of CPU efficiency with increased load on Grid Canada.

### 2.4.6 Conclusion of Results

It is apparant that the GridFTP method is the fastest and most efficient method by which to run the ATLAS simulation over the grid. The Full and Partial AFS method's poor performance can be blamed on AFS bottlenecks. The Full GridFTP method's excellent performance can be attributed to its minimal use of AFS; only for access to the ATLAS software. Further experimentation and analysis was done to determine the Full GridFTP method's performance with increasing load on the grid testbed.

## 2.5 Further Testing of The Full GridFTP Method

The Full GridFTP method was tested with one, five, ten, and fourteen jobs running simultaneously on the Grid Canada testbed. Three remote resources were observed throughout these tests: tree.pfc.forestry.ca at the Pacific Forestry Centre in Victoria, BC, Canada, thuner-gw.phys.ualberta.ca at the University of Alberta in Edmonton, Alberta, Canada, and ontario.iit.nrc.ca at NRC in Ottawa, Ontario, Canada. The total runtime, the CPU efficiency and the CPU utilization were observed to determine the performance. The CPU utilization is defined as:

$$U = T_{real}/TotalRuntime \tag{2}$$

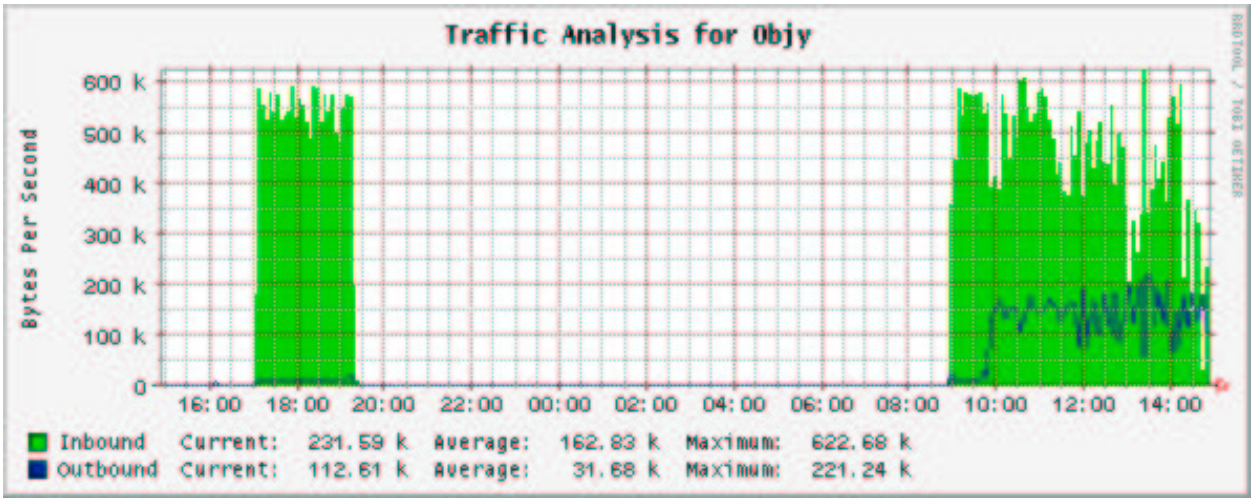


Figure 5: Triple job submission using the Partial AFS method took almost 5 hours due to AFS bottlenecks

The CPU utilization is a measure of the portion of time the remote CPU spends running the ATLAS simulation. This does not include the times for transferring, tarring, or untarring of data. A remote resource with a CPU use of around 50% is considered to be well utilized. The results from these tests are presented in tabular format in Appendix B.

### 2.5.1 Staggered Job Submission

Although this method of ATLAS simulation is not AFS intensive, increased job submissions do put considerable load on the local machine, in this case chimera.phys.uvic.ca at UVic. When three jobs were run using the Full GridFTP, as in the previous section, as opposed to a single job the total runtime increased over two fold. The two fold increase in runtime was mainly due to data transfer bottlenecks at chimera caused by the transfer of input and output data. Instead of sending over the input data to a machine in under 8 minutes, the input data was sent to three machines over the course of 20 minutes. If the remote machines complete the ATLAS simulation at the same time, then three sets of the output data are sent back to the local machine at the same time causing another bottleneck. Increased total runtime caused by data transmission bottlenecks greatly decreases the CPU utilization of a remote resource.

A solution to prevent bottlenecks at the local machine is to stagger the ATLAS job starts by the average amount of time it takes to send the input data to an execution machine. Through experimentation it was concluded that the input data can be sent under 8 minutes to any remote machine under daytime network loads. The delay time for job submission was then chosen to be ten minutes to allow for any network inconsistencies. All of the tests discussed in the remainder of this report were run using staggered starts.

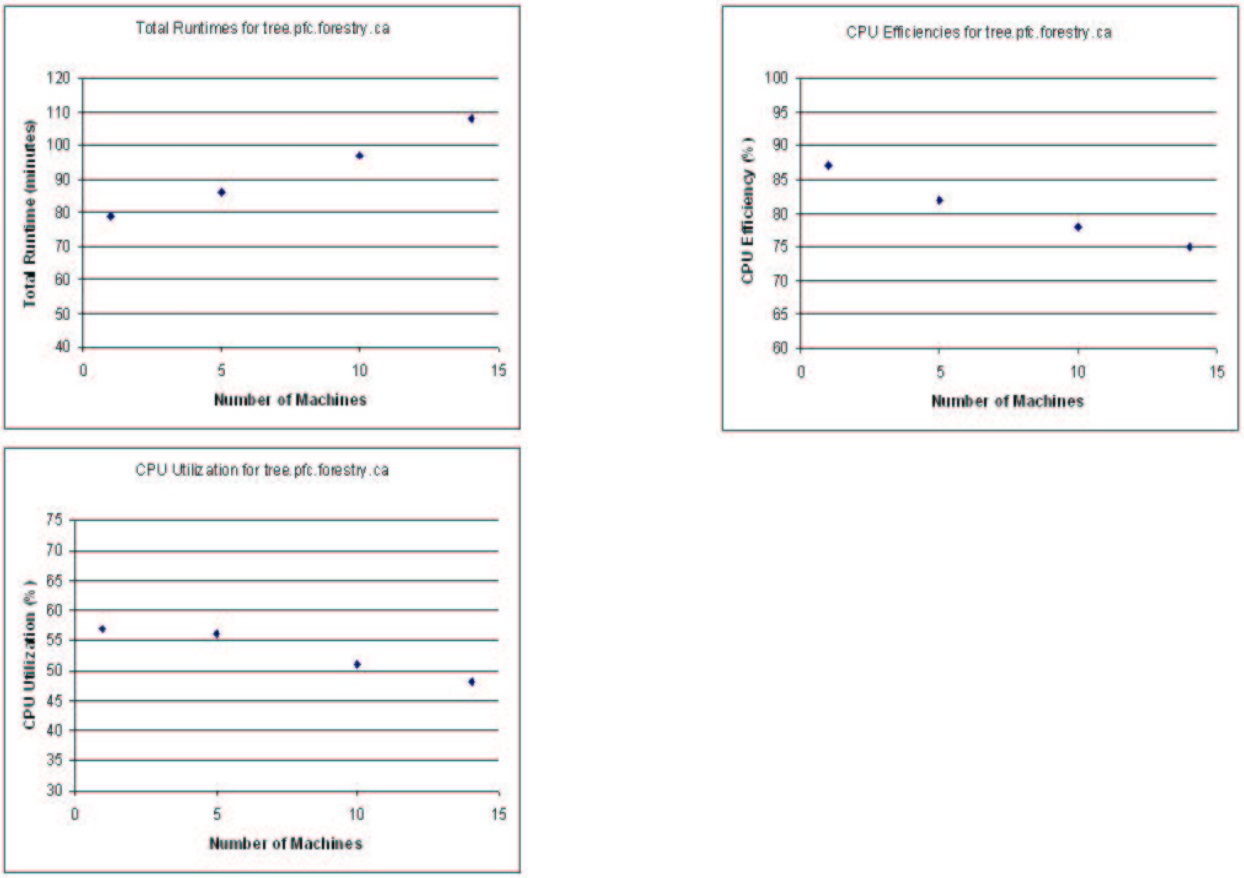


Figure 6: Tree.pfc.forestry.ca showed the poorest performance for the Full GridFTP tests

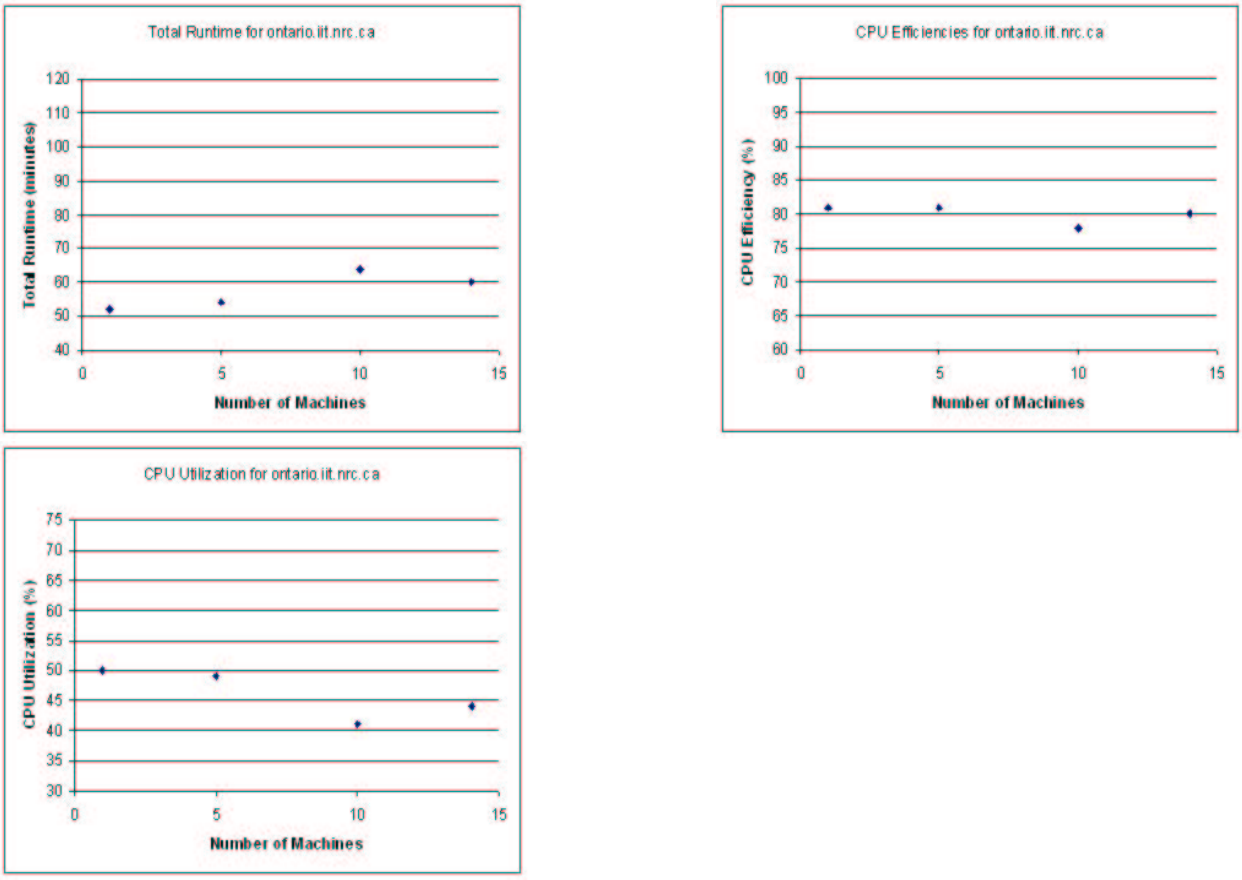


Figure 7: Ontario's performance was good except for its CPU utilization

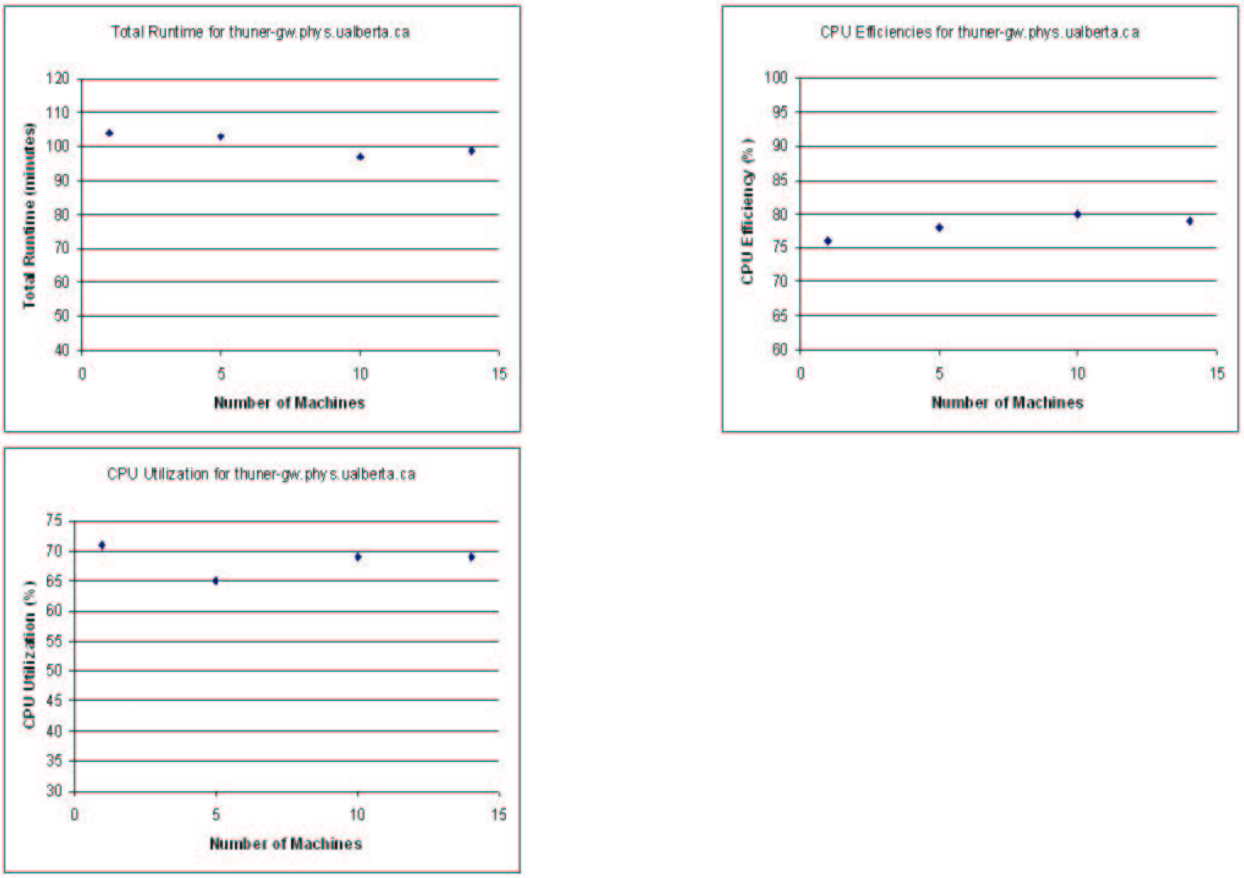


Figure 8: Thuner showed the best performance throughout the tests

### 2.5.2 Total Runtime Results

As an increasing number of ATLAS jobs were submitted to the grid simultaneously, the total runtime of three of the jobs was observed. Figures 6, 7, and 8 show the runtimes for tree.pfc.forestry.ca, ontario.iit.nrc.ca, and thuner-gw.ualberta.ca respectively. The runtimes ranged from 52 minutes to 1 hour and 48 minutes. It is hard to draw any conclusions from the runtime plots about the behaviour of total runtime with increasing job submission.

Tree's runtime graph in figure 6 shows an increase in runtime of 30 minutes from about 80 minutes with a single job submission to 110 minutes for fourteen simultaneous submissions. However, ontario.iit.nrc.ca's maximum runtime is observed at ten job submissions with a difference of only 13 minutes from a single submission. Furthermore, thuner-gw.phys.ualberta.ca's runtime shows a small decrease in total runtime as more jobs were submitted.

### 2.5.3 CPU Efficiency Results

The CPU efficiency for each machine was also observed as an increasing number of jobs were submitted to the grid. Figures 6, 7, and 8 also show the CPU efficiency trends. The efficiencies range from 75% to 87%. PFC's machine, tree.pfc.forestry.ca, showed a steady decrease in efficiency from a maximum of 87% to a minimum of 75%. However, ontario.nrc.iit.ca's machine kept a relatively constant efficiency with a low of 78% and a high of 81%. Thuner-gw.phys.ualberta.ca again showed a suprising result by increasing its CPU efficiency up to ten simultaneous jobs and then dropping off slightly at fourteen jobs.

### 2.5.4 CPU Utilization Results

Figures 6, 7, and 8 also show the CPU utilization for each machine with increasing load on the grid. The utilizations ranged from a low of 41% to a high of 71%. All three machines showed a decrease in CPU utilization as more jobs were submitted. Tree.pfc.forestry.ca showed a steady decrease while thuner and ontario showed irregular trends. Ontario's minimum, 41%, occured at 10 jobs with a utilization of 44% for 14 jobs. Thuner's minimum utilization, 65%, occured at five jobs and stayed constant for ten and fifteen jobs at 69%.

### 2.5.5 Discussion of Total Runtimes

One would predict that as more jobs are submitted to the grid, the total runtime for each machine would increase. However, this result is not obvious by analyzing the results of the runtime tests. Suprisingly, University of Alberta's machine, thuner-gw, completed the ATLAS jobs quicker as more jobs were simultaneously submitted. Ontario's runtime did increase but only by 13 minutes.

Tree showed the worst results with an increase of 30 minutes. This is mainly due to the poor network connectivity between UVic and the Pacific Forestry Centre. According to the table in Appendix B the input data transfer times to Tree were much larger than transfer times to Thuner-gw or Ontario. PFC and UVic will soon be connected via the Victoria Transit Exchange, which will greatly improve connectivity between UVic and PFC. Overall, the total job runtimes via the GridFTP method are acceptable.

### 2.5.6 Discussion of CPU Efficiencies

Ontario and thuner-gw showed quite promising results for CPU efficiency. Ontario kept relatively steady around 80% while thuner-gw's efficiency actually increased. The CPU efficiency can be viewed as a measure of how long a CPU is waiting for input inorder to continue computing. A steady high efficiency proves that the software on AFS is easily distributed across the grid to remote sites with little or no lag in performance due to added load.

Because ontario and thuner-gw showed remarkable performance, Tree's poor behaviour can be attributed to the poor network connectivity between UVic and PFC, and not the performance of AFS. Overall, the CPU effciency held up well under increased load across the Grid Canada testbed.

### 2.5.7 Discussion of CPU Utilization

As a rule of thumb, a utilization of 50% is considered acceptable. Most of the results from the previous tests gave CPU utilization's above 50%. As more jobs were submitted simulataneously, all three machines experienced decreasing utilization. ontario and tree lagged under 50%. Ontario ran at a utilization of 44% while tree ran at 48% for fourteen jobs.

The CPU utilization is the most important factor used in determining the performance of the grid. It compares the time spent computing, which cannot be changed, to the time spent tranferring files and performing untarring and tarrring functions. The time spent transferring files can be improved via better network connectivity across the grid, thus increasing CPU utilization. The GridFTP method for simulation works well for a few jobs submissions but tends to lag for many submissions. Improved network connectivity will give better results.

# 3 Conclusions

The GridFTP method proved to be a successful means by which to run multiple ATLAS simulations across the Grid Canada testbed. The merit of this method was measured using three parameters: Total runtime, CPU efficiency, and CPU utilization. The total runtimes and CPU efficiencies stood at acceptable levels proving that AFS is a reliable means by which to obtain the ATLAS software. The CPU utilization began to lag under 50% as fourteen simulataneous jobs were run. Better network connectivity will speed up data transfer times, and in turn higher CPU utilizations will be experienced.

The GridFTP method was also tested against two other more convenient methods: the Full AFS and the Partial AFS methods. Ridiculously long runtimes along with extremely low CPU efficiencies removed the Full AFS method as a candidate for ATLAS simulation. The Partial AFS method showed more promise than the Full AFS method, however it showed poor performance when more than one job was simultaneously run across the grid testbed.

The ATLAS simulation, used to predict the performance and aid in the design of the ATLAS detector, was not originally written to run in a distributed computing environment. The simulation has put the Grid Canada testbed to the test, providing physicists with the

opportunity to learn more about grid computing and its application to high energy physics. Recent tests upon Grid Canada have given the UVic grid team a better understanding of grid technologies and their shortcomings.

# 4 Recommendations

More tests using the GridFTP method with increased network connectivity to the Grid Canada testbed from UVic will most likely yield improved results. In the next month UVic will be properly connected to the Victoria Transit Exchange which provides almost a direct link to CA*net 4, the Canadian light backbone. More than three computers across the grid testbed should be observed as a higher number of simulataneous jobs are submitted over the faster network. Observing more machines will provide greater insight into the perfomance of the grid and give a better understanding of the ATLAS simulation's impact on the network and AFS.

To decrease data transfer times, the ATLAS input data should be stored remotely on storage facilities across the country. The ATLAS scripts will then be modified to obtain the input data from the closest, or minimum latency, server available. These facilities could also store the output data which could be downloaded when needed. Smaller input and output data transfer times will in turn increase CPU utilization and decrease total runtimes across the grid.

# References


[1] T. Curniski, What is Grid Computing, Grid Computing (May 2002) 1-6.

[2] D. Vanderster, December 20, 2002 Grid Computing Using Particle Physics Applications, *grid.phys.uvic.ca/docs/dvwtr.pdf* (2002)

[3] Unknown, Powering up the Grid, BBC News, *news.bbc.co.uk/hi/english/sci/tech/newsid_806000/806410.stm* (June 2000)

[4] R. Sobie, UVic Grid Testbed Homepage, *grid.phys.uvic.ca* (2002).

[5] I. Foster, The Globus Project: Frequently Asked Questions, *www.globus.org/about/faq/general.html* (2003)

[6] I. Foster, Globus 2.2 Services, *www.globus.org/gt2.2/admin/guide-services.html* (2003)

[7] R. Hatem, CA*net 4, *www.canarie.ca/canet4/* (2003)

# A The ATLAS Scripts

## A.1 How is the ATLAS Simulation Run?

There are a few different ways to run the ATLAS simulation. In all cases the software for the simulation is read off the AFS tree. The different cases are:
1. *The Full AFS Method:* Run the simulation with the input data on AFS. The output is written directly to AFS throughout execution.
2. *The Partial AFS Method:* The input data is sent over to the remote machine via gridFTP. As the simulation runs the output is written to the AFS tree.
3. *The Full GridFTP Method:* The input data is sent over to the execution machine. The output data is stored remotely during execution. After the simulation is complete, the output is sent back to UVic via gridFTP.
4. *The Partial GridFTP Method:* The input data is assumed to already exist on the remote machine. This method is used if the second cpu of a dual machine is required. Once the simulation is complete the output data is sent back to UVic via gridFTP.
5. *The Grid Cluster Method:* Run the simulation on the UVic grid cluster with the software on AFS. The input data is stored on each of the gridnodes and the output is written to an NFS mounted directory.

The ATLAS input data consists of ten files. These files have a .zebra extension. The output data consists of one file by the name of dc1.002099.lumi10.00001.hlt.pyt_min_bias.zebra.

## A.2 The Four different kinds of scripts

There are five different types of scripts that are used for the ATLAS simulation:
1. *The executable script:* 'dc1.pileup.standard.v2', is responsible for actually running the atlsim.exe executable. This script is made by the ATLAS software developers and is not to be modified. This script is unique and is available at /afs/phys.uvic.ca/atlas-write/4.0.1
2. *The wrapper scripts:* Responsible for setting path variables. These scripts decide where the software and input data is read from, and where the output data will be stored during execution. These scripts are written and modified by people who wish to run the ATLAS simulation via different methods. These scripts are also available at /afs/phys.uvic.ca/atlas-write/4.0.1 and have 'dc1.pileup.standard.v2.wrap4' in their title.
3. *The remote scripts:* Executed on the remote site. These scripts take care of making the remote directories, logging onto AFS, and other procedures pertinent to running a job remotely. There are five different remote script: runAtlas_remote.sh, runAtlas_remoteSecondCPU.sh, runAtlas_remoteUVIC.sh, runAtlas_remoteAFS.sh, and runAtlas_remoteAFSPartial.sh
4. *The local scripts:* Executed on a local machine. There are three different local scripts: runAtlas.sh, runAtlas_SecondCPU.sh, and runAtlas_AFSPartial.sh. They call runAtlas_remote.sh, runAtlas_remoteSecond CPU, and runAtlas_remoteAFSPartial.sh respectively. The local scripts are responsible for sending over the input data, running the necessary remote scripts on the remote machine and taking care of the output data.
5. *The Batch Scripts:* Executed on a local machine. These scripts encompass local and remote scripts to send out multiple atlas jobs to multiple resources.

There is also a script named 'makeAtlasdirectories.sh' this script is sent over to the execution machine when using the Full GridFTP and Partial AFS methods. All of the scripts that deal with sending input data assume that one tarred and compressed file, containing all of the input data files exists.

## A.3 Running ATLAS Via The Full GridFTP Method

This method uses runAtlas.sh, runAtlas_remote.sh and makeAtlasdirectories.sh. The steps involved in running a job this way are:
1. Tarred up and compressed input data is sent over to the execution (remote) machine from the local machine via gridFTP (the input data, inputdata.tgz, exists in the /home/manj/002099 directory on chimera.phys.uvic.ca).
2. Input data is untarred, uncompressed on remote machine.
3. The simulation is run.
4. Output data is tarred,compressed and saved in a different folder from where the output was written. The original output is removed from the remote machine.
5. There are 3 modes by which the output data can be transported back to the local machine:
5a.*Mode 1:*
- Output data sent via gridFTP to AFS.
- Ouput data untarred and uncompressed on AFS.
5b.*Mode 2:*
- Output data sent via gridFTP to the local machine.
- Then sent to AFS.
- Then untarred and uncompressed on AFS.
5c.*Mode 3:*
- Output data sent via gridFTP to local machine.
- Untarred and uncompressed on local machine.
- Untarred and uncompressed data sent via gridFTP to AFS.

### A.3.1 runAtlas.sh

runAtlas.sh is called on the command line and takes three input parameters:
- *HOST:* The name of the remote machine that you wish to execute upon.
- *JOBNAME:* Name of the job in the format 'MonthDay_RemoteMachineName'.
- *REMOTEHOME:* Home directory on the remote machine.
runAtlas.sh has seven user variables:
1. *HOME:* The root directory on the submission(local) machine.
2. *INPUTDIR:* The directory, on the remote and local machines where the input data will be.
3. *TEMPDIR:* The output data and log file will be placed in this directory.
4. *OUTPUT:* The name of the compressed and tarred output data.
5. *UNCOMPRESSEDOUTPUT:* The name of the uncompressed and untarred output data.
6. *MODE:* The mode used for output data transfer, as described above.

7. *STREAMS:* The number of streams used in the various data transfers with gridFTP.

### A.3.2 runAtlas_remote.sh

runAtlas_remote.sh is called in the runAtlas.sh script. runAtlas.sh passes JOBNAME, OUTPUT and REMOTEHOME as input parameters. This script runs on the remote machine. It first obtains AFS tokens to access the ATLAS software. The software exists on the AFS tree. Second, it runs the dc1.pileup.standard.v2. wrap4_inout script, which exists on the AFS tree. dc1.pileup.standard.v2.wrap4_inout is a wrapper script that sets up the paths for input and output of the Atlas simulation. The wrapper script calls dc1.pileup.standard.v2, which runs the actual simulation. Lastly, runAtlas_remote.sh tars up and compresses the output data and saves it to the same location as the input data. The file dc1.002099.lumi10.00001.hlt.pyt_min_bias. zebra, the output data, is removed from the remote machine. At this point runAtlas.sh takes control again and transfers the tarred and compressed output data to the local machine.

Here is a sample command to submit one of these jobs. Be aware that all of the remote and local scripts and makeAtlasdirectories.sh need to be in the same directory:

*time ./runAtlas.sh ontario.iit.nrc.ca mar31_ontario home/agarwal*

The time command will give the total time of execution of the job. Time commands are also inserted throughout the scripts. As the job runs, the time for each step of the job will be shown on the terminal output.

## A.4 Running ATLAS Via The Partial GridFTP Method

This method is used if the correct directories and the input data already exists on the remote machine; Assuming that the input data has already been untarred and uncompressed. This method enables the use of a second CPU on a remote machine. If the first CPU already has an atlas job running on it, then submitting a Partial GridFTP job will use the next avaliable CPU along with the input data that already exists on the remote machine. The Partial GridFTP Method uses a similar procedure to the Full GridFTP Method, excluding the first two steps.

Calling a partial gridFTP job is very similar to submitting a full GridFTP job. However makeAtlasdirectories is not needed:

*time ./runAtlas_SecondCPU.sh ontario.iit.nrc.ca mar31_ontario home/agarwal*

## A.5 Running ATLAS Via The Full AFS Method

This method uses runAtlas_remoteAFS.sh. Unlike runAtlas_remote.sh, runAtlas_remoteAFS.sh takes only one parameter, JOBNAME, and gets called on the command line.

With this method the remote machine is used for processing power only. No data pertinent to the ATLAS simulation is stored on the remote machine. The input data is kept on the

AFS tree at /afs/phys.uvic.ca/atlas/4.0.1/project/dc1/simul/data/002099/ and the output data is written to /afs/ phys.uvic.ca/atlas-write/4.0.1/temp/JOBNAME/dc1.002099.lumi10.00001.hlt.pyt_min_bias/.

The runAtlas_remoteAFS script is run on the remote machine and is responsible for getting the AFS permissions and calling the atlas wrapper script, dc1.pileup.standard.v2.wrap4. This wrapper script sets up all input and output paths to AFS. It is also responsible for invoking dc1.pileup.standard.v2, which takes care of running the simulation.

Here is an example for running a full AFS job:

*time globus-job-run ontario.iit.nrc.ca -s runAtlas_remoteAFS.sh mar31_ontario home/agarwal*

The above line uses a Globus command directly. The -s option tells the command to stage runAtlas_remote.sh. Stage means to send the script to the remote machine for execution.

## A.6 Running Atlas Via the Partial AFS Method

This method uses runAtlas_AFSPartial.sh, runAtlas_remoteAFSPartial.sh and makeAtlasdirectories.sh. The steps involved in running a job this way are:
1. Tarred up and compressed input data is sent over to the execution (remote) machine from the local machine via gridFTP (the input data, inputdata.tgz, exists in the /home/manj/002099 directory on chimera.phys.uvic.ca).
2. Input data is untarred, uncompressed on remote machine
3. Ouput is written to AFS as the simulation is run

The script runAtlas_AFSPartial takes the same input parameters as runAtlas.sh: HOST, JOBNAME, and REMOTEHOME. Similar to runAtlas.sh, runAtlas_AFSPartial also defines HOME, INPUTDIR, and STREAMS. However, it does not need to define any variables relevant to output control. The output is taken care of via AFS. Here is an example for running a partial AFS job:

*time ./runAtlas_remoteAFSPartial.sh ontario.iit.nrc.ca mar31_ontario home/agarwal*

## A.7 Running Atlas Via the Grid Cluster Method

The grid cluster consists of a gatekeeper machine, grid.phys.uvic.ca, and 5 grid nodes. The grid nodes are behind the firewall (grid.phys.uvic.ca). If you want to run an Atlas job on one of the UVic grid nodes, you need to use runAtlas_remoteUVIC.sh.

A copy of the input data for the ATLAS job is located on each grid nodes harddrive in the /atlas/002099/ directory. When the simulation is running, the output data is written on the /home/agarwal. The /home/agarwal directory is located on grid.phys.uvic.ca and is NFS mounted on each grid node.

Here is an example for running a UVic cluster job:

*time globus-job-run grid.phys.uvic.ca/condor -s runAtlas_remoteUVIC.sh mar31_grid1 home/agarwal*

Condor is the job manager on grid.phys.uvic.ca. It takes the job and gives it to the next available grid node. If grid1 is busy the job will be executed on grid2, if grid2 is busy the job will be executed on grid3 and so on.

## A.8 The Batch Scripts

To exploit grid resources across the country to the fullest, multiple jobs need to be running at the same time. The scripts described so far only run a single simulation at a single resource. Two batch scripts have been written to submit many jobs to various resources; These scripts are: runAtlasBatchRemote and runAtlasBatchUVIC.

The script runAtlasBatchUVIC sends six jobs to the UVic grid cluster to use up all available CPUs. The script runAtlasBatchRemote sends jobs out to a variety of remote resources aswell as the UVic grid cluster via runAtlasBatchUVic. To run a batch job, you only need to execute runAtlasBatchRemote. Running runAtlasBatchRemote submits jobs using the Full GridFTP, Partial GridFTP, and UVic Grid Cluster methods. Here is an example of running a batch job:

*./runAtlasBatchRemote*

# B GridFTP Test Results

Starting on April 14th 2003, a series of tests were performed to test the merit of the Full GridFTP method for ATLAS simulation. Three computers, tree.pfc.forestry.ca, ontario.iit.nrc.ca, and thuner-gw.phys.ualberta.ca were observed as one, five, ten and fourteen jobs were submitted to the Grid Canada testbed simultaneously. The table below uses acronyms to show the time taken for each step of the remote execution of an ATLAS job.

1. T.I: Time to transfer tarred and compressed input to remote site
2. Utar: Time to untar and uncompress the input data on remote site
3. R: Real time of the remote CPU when executing the ATLAS simulation

4 U: User time of the remote CPU when executing the ATLAS simulation

5. S: System time of the remote CPU when executing the ATLAS simulation
6. Tar. O: Time to tar and compress the output data from the simulation
7. T.O: Time to transfer the tarred and compress output back to the local machine
8. Utar. O: Time to untar and uncompress the output data on local machine
9. Tot: Total time of running remote Atlas job
10. Eff: The CPU efficiency of the ATLAS simulation
11. Util: The CPU utilization of the ATLAS simulation

| Machine | T.I | Utar | R | U | S | Tar.O | T.O | Utar.O | Tot | Eff | Util |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Single Job | | | | | | | | | | | |
| Tree | 10:42 | 4:27 | 45:28 | 38:11 | 1:16 | 8:32 | 5:29 | 4:11 | 79:48 | 87% | 57% |
| Ontario | 5:34 | 3:17 | 26:22 | 20:33 | :58 | 6:49 | 6:30 | 3:58 | 52:35 | 81% | 50% |
| Thuner | 6:47 | 8:24 | 73:40 | 48:31 | 7:23 | 8:10 | 2:05 | 4:20 | 103:55 | 76% | 71% |
| Five Jobs | | | | | | | | | | | |
| Tree | 7:37 | 4:24 | 47:53 | 37:54 | 1:15 | 5:59 | 9:23 | 9:44 | 86 | 82% | 56% |
| Ontario | 6:20 | 3:18 | 26:28 | 20:28 | 0:59 | 6:48 | 6:24 | 4:19 | 54:07 | 81% | 49% |
| Thuner | 8:02 | 7:42 | 67:40 | 47:38 | 4:56 | 8:36 | 2:31 | 7:50 | 103:21 | 78% | 65% |
| Ten Jobs | | | | | | | | | | | |
| Tree | 13:18 | 4:27 | 50:15 | 37:59 | 1:14 | 6:21 | 5:22 | 16:44 | 97:35 | 78% | 51% |
| Ontario | 7:47 | 3:16 | 26:32 | 20:35 | 1:0 | 6:47 | 3:22 | 16:33 | 64:43 | 78% | 41% |
| Thuner | 7:57 | 6:48 | 66:28 | 47:48 | 5:30 | 8:34 | 2:48 | 3:59 | 96:53 | 80% | 69% |
| Fourteen Jobs | | | | | | | | | | | |
| Tree | 22:32 | 4:28 | 52:17 | 37:48 | 1:15 | 6:30 | 6:35 | 12:41 | 108:41 | 75% | 48% |
| Ontario | 5:47 | 3:27 | 26:27 | 20:32 | 0:58 | 6:46 | 4:13 | 13:04 | 60 | 80% | 44% |
| Thuner | 7:39 | 7:18 | 68:36 | 48:05 | 6:03 | 8:45 | 2:03 | 4:38 | 99:22 | 79% | 69% |

Table 3: The time for each step of the Full GridFTP method is shown with an increasing number of jobs