UVIC Grid Testbed: Modifications to the phys.uvic.ca AFS Cell

UVIC Grid Testbed and Grid Canada
Department of Physics and Astronomy
University of Victoria
Victoria, British Columbia

Jenny Allan
0021379
Computer Engineering
jjallan@engr.uvic.ca

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
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.

April 23, 2003

Contents

1 Introduction **1**

1.1 Grid Computing 1

1.2 Grid Canada and the UVIC Grid Testbed 1

2 ATLAS Tests on Grid Canada, Fall 2002 **2**

2.1 Setup of AFS, Fall 2002 2

2.2 ATLAS event simulation jobs, Fall 2002 2

3 Intended Usage of AFS, Spring 2003 **6**

3.1 ATLAS event simulation jobs, Spring 2003 6

4 Proposed changes to AFS to Increase Efficiency **7**

4.1 Clones and Replicated Volumes 7

4.2 The submitted proposal 8

5 Accepted Setup of AFS, Spring 2003 **8**

6 Results and Conclusions **9**

6.1 Seperate Submission to Single Remote Hosts 9

6.2 Simultaneous Submission on Two Remote Hosts 12

6.3 Simultaneous Submission to Multiple Remote Hosts 16

7 Conclusion **21**

8 Recommendations **21**

A AFS: A Distributed File System **1**

A.1 Servers and Clients 1

A.2 Cells 1
A.3 Transparent Access and the Uniform Namespace 2
A.4 Volumes 2
A.5 Mount Points 2
A.5.1 The Three types of Mount Points 3
A.5.2 Rules of Mount Point Traversals 5
A.6 Efficiency Boosters: Replication and Caching 6
A.7 Security: Mutual Authentication and Access Control Lists . . 7
A.8 Details of Changes made to the phys.uvic.ca AFS cell 7

B The Globus Toolkit **1**
B.1 User Commands 1

List of Tables

1 Basilisk.phys.uvic.ca Hardware/Software Information 4

2 Past Volume Mount Points on the phys.uvic.ca AFS cell . . . 5

3 Objy.phys.uvic.ca Hardware/Software Information 8

4 Current Volume Mount Points on basilisk.phys.uvic.ca 10

5 Current Volume Mount Points on objy.phys.uvic.ca 11

6 Job Submission times for **vision.triumf.ca** by itself 11

7 Job Submission times for **thuner-gw.phys.ualberta.ca** by itself 12

8 Job Submission times for **vision.triumf.ca** at the same time as thuner-gw.phys.ualberta.ca 13

9 Job Submission times for **thuner-gw.phys.ualberta.ca** at the same time as vision.triumf.ca 15

10 Job Submission times for **vision.triumf.ca**, when submitted to a total of six grid resource sites simultaneously. 18

11 Job Submission times for **thuner-gw.phys.ualberta.ca**, when submitted to a total of six grid resource sites simultaneously. 18

List of Figures

1 UVIC Grid Testbed 3

2 **Single Hosts Without Clones - Seperate Submission** . 13

3 **Single Hosts With Clones - Seperate Submission** . . . 14

4 **Single Hosts Without Clones - Simultaneous Submission** 16

5 **Single Hosts With Clones - Simultaneous Submission** 17

6 **Multiple Hosts Without Clones - Simultaneous Submission** 19

7 **Multiple Hosts With Clones - Simultaneous Submission** 20

Abstract

This report discusses improvements to the phys.uvic.ca AFS cell at the UVIC Grid Testbed. During the fall of 2002, BaBar and ATLAS tests were conducted over the UVIC Grid Testbed and Grid Canada by a previous co-op student, Dan Vanderster. It was found during those tests that a main source of the low usage of the remote host CPU cycles was due to latencies incurred over the network during the numerous interactions with the phys.uvic.ca AFS cell. These interactions occurred when the remote hosts sourced input data and execution scripts from and sent output data to the phys.uvic.ca AFS cell in Victoria. A recommendation was to improve the job submission method by using Grid-FTP to transfer the input and output data files and only use AFS to source the execution scripts. Since the execution scripts are static it was suggested to turn them into a read-only volume and to create clones in order to increase efficiency. AFS has a faster, more efficient method of reading clones. If a volume is read-only (a clone or a replicated volume) then the client only has to contact the AFS server once to cache the volume. However, if a volume is read/write (not static) then the client has to contact the AFS server once to cache the volume, then each time it opens a file it has to contact the AFS server to ensure that the file has not been altered since it was cached. Plus, by adding more sites in the phys.uvic.ca AFS cell, the data becomes more accessible simply due to the fact that there are more places from which it can be sourced.

Another machine was added to the phys.uvic.ca AFS cell and two clone/replicated volumes were created. Various tests were conducted without the clones and then again with the clones. It was obvious from the differences in remote host CPU efficiencies, that the advent of the clones did, in fact, increase the efficieny of the phys.uvic.ca AFS cell.

1 Introduction

This report discusses the attempts and results of those attempts to increase the efficiency of the phys.uvic.ca *AFS* (Andrew's File System) cell at the UVIC Grid Testbed. For a proper introduction to AFS please see Appendix A. Before jumping to the focus of this report: AFS Modifications, the term Grid Computing will be defined and the organizations involved in the research discussed in this report will be identified.

1.1 Grid Computing

Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations. Grid technologies promise to change the way organizations tackle complex computational problems [1]. The concept of Grid Computing was defined by Ian Foster, et al., one of the founding fathers of grid computing and the Globus Project. In his paper *The Anatomy of the Grid* [2], grid computing is said to be "coordinated research sharing and problem solving in dynamic, multi-institutional virtual organizations". The Globus Toolkit is a product of the Globus Project, initiated in 1996 at Argonne National Laboratory, U.S.A. [3]. This open source software enables secure usage of distributed, heterogeneous domains using open protocols, interfaces, and schemas. For more information about Globus please see Appendix B.

1.2 Grid Canada and the UVIC Grid Testbed

Grid Canada and nearly all other grids around the world are implemented with the Globus Toolkit. Grid Canada is an organization created in partnership between *CANARIE* [4], *C3.ca* [5] and the *National Resarch Council of Canada* [6]. For more information about Grid Canada, please see [7]. The UVIC Grid Testbed is located in the Department of Physics and Astronomy at the University of Victoria. The professional researchers in the Department of Physics and Astronomy at the University of Victoria started the UVIC Grid Testbed. It was established to research Grid solutions that will be required in order to be able to access the computational power needed to simulate HEP (High Energy Physics) events and process the pedabytes of data from the *ATLAS* experiment at *CERN* (European Organization

for Nuclear Research), which is their main research focus [8]. For more information about ATLAS and/or CERN please see [9] and [10]. The people responsible for the maintenance and management of the UVIC Grid Testbed are also partially or fully responsible for some of the other grid resources in Grid Canada. The UVic Grid Testbed consists of a cluster of five Condor [11] controlled, Redhat 7.2 Linux machines behind a firewall, accessed through a Globus gateway, plus the phys.uvic.ca AFS cell, which consists of two server machines. The UVIC Testbed can be seen in Figure 1. The UVIC Grid Testbed is used as a local and initial grid resource to test applications that are to be used on Grid Canada. The UVIC Grid Testbed's current project is to research ways in which to exploit grid resources (CPU cycles) from HEP and non-HEP organizations throughout Canada, with minimal inconvenience to the owners of those resources.

2 ATLAS Tests on Grid Canada, Fall 2002

2.1 Setup of AFS, Fall 2002

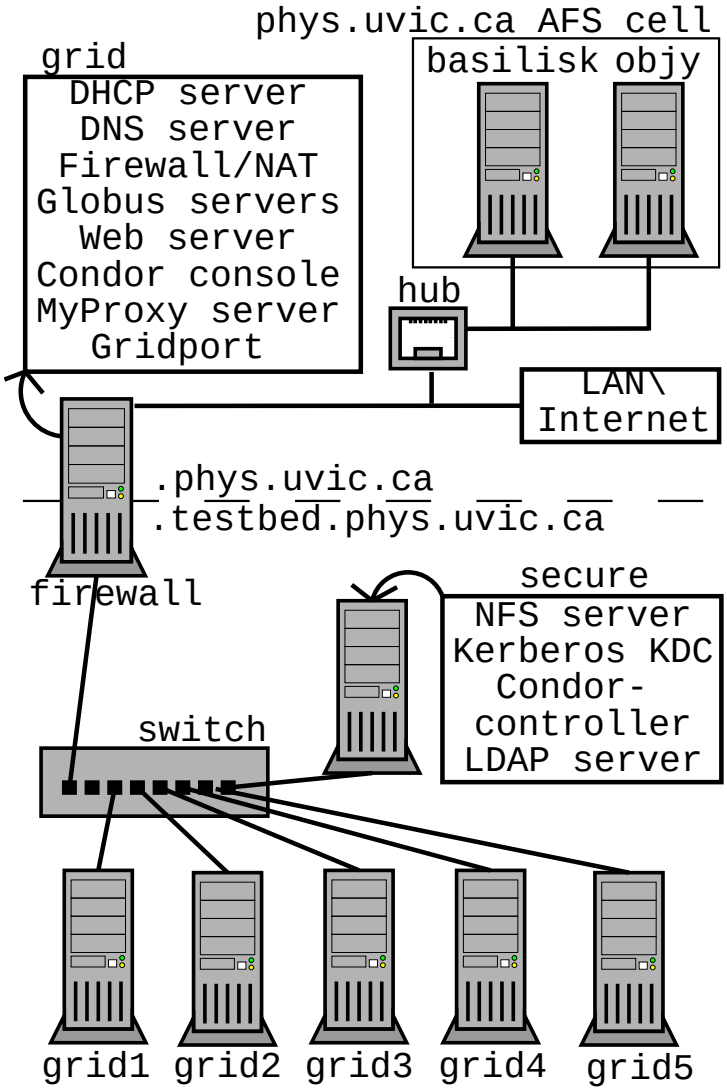
During the fall of 2002, UVIC Testbed Co-op student Dan Vanderster ran ATLAS 4.0.1 data challenge tests. Globus 2.0 was used as the platform upon which the tests were run and AFS was used as the intermediary for data exchange. During that period the AFS setup at UVIC consisted of the one machine, basilisk.phys.uvic.ca, running as the Database Server, File Server, Binary Distribution, and System Control Machines for the phys.uvic.ca AFS cell. Some of its hardware/software information can be seen in Table 1. The volumes within the phys.uvic.ca AFS cell and their respective mount points can be seen in Table 2. Note that all the mount points shown in Table 2 are regular mount points. See Appendix A, for more information on mount point traversals.

2.2 ATLAS event simulation jobs - submitted using Globus and AFS, Fall 2002

The ATLAS event simulation jobs were submitted via globus, as follows:

1. Using globus-job-run, a Perl executable script was submitted to a remote grid host.

Figure 1: UVIC Grid Testbed



2. Remote host fetches ATLAS input data and ATLAS .zsh event simulation scripts off the phys.uvic.ca AFS cell.

3. Execute the ATLAS job on the remote host.

4. As the job is executing, output data is sent to the phys.uvic.ca AFS cell.

On average these jobs took 39.62 hours to complete. It was observed that the input data was constantly read in to the foreign host from the AFS server. This is because the local AFS cache on the foreign machines is smaller then the large ATLAS input files (total size of ATLAS input files is 2.5 GB). Since the remote host could not read in the input files as fast as the input files could be processed, a CPU efficiency of only 10% on average was observed. Latency increased when the number of jobs being run simultaneously was increased. This is because the number of resources trying to fetch input files and ATLAS executable from the AFS server cause a traffic jam. The latencies caused the CPU efficiency to decrease. It was also observed that as the local AFS cache became full, output data was sent to the AFS server in surges, rather than constantly. This does not efficiently make use of the AFS server nor does it properly use the CPU cycles of the remote grid resource. One of the recommendations made after these tests was to try sending the data to the remote host via a more efficient method than AFS. Execute the ATLAS job remotely using globus and AFS to fetch the ATLAS executable from AFS. Send all output data to local disk. Then, send the output files (total size of output files 2.8 GB) back to UVIC (and eventually AFS), after the ATLAS execution is complete [12]. These recommendations are being currently pursued (spring of 2003).

Table 1: Basilisk.phys.uvic.ca Hardware/Software Information

MODEL	Pentium MMX	CPU	
RAM	128MB	Processor:	Single
OS	RedHat Linux 7.3 (valhalla)	Speed:	200MHz
DISKSPACE			
hda	20GB	Maxtor	5T02H2
hdb	56GB	Maxtor	6L060J3
hdc	56GB	Maxtor	6L060J3

Table 2: Past Volume Mount Points on the phys.uvic.ca AFS cell

VOLUME NAME	MOUNT POINT
vicepa	
public	/afs/phys.uvic.ca/public
root.afs	/afs
root.cell	/afs/phys.uvic.ca
vicepb	
home.babar.kanga	/afs/phys.uvic.ca/babar/kanga
home.gable	/afs/phys.uvic.ca/home/gable
home.gabriel	/afs/phys.uvic.ca/home/gabriel
home.gking	/afs/phys.uvic.ca/home/gking
home.jjallan	/afs/phys.uvic.ca/home/jjallan
home.manj	/afs/phys.uvic.ca/home/manj
home.sobie	/afs/phys.uvic.ca/home/sobie
home.wyvern	/afs/phys.uvic.ca/home/wyvern
home.zwiers	/afs/phys.uvic.ca/home/zwiers
vicepc	
home.babar	/afs/phys.uvic.ca/babar
home.babar.cern	/afs/phys.uvic.ca/babar/cern
home.babar.objectivity	/afs/phys.uvic.ca/babar/package/objectivity
home.babar.package	/afs/phys.uvic.ca/babar/package
home.babar.packages	/afs/phys.uvic.ca/babar/dist/packages
home.babar.releases	/afs/phys.uvic.ca/babar/dist/releases
home.babar.roguewave	/afs/phys.uvic.ca/babar/package/RogueWave
home.dvanders	/afs/phys.uvic.ca/home/dvanders
home.nkanaya	/afs/phys.uvic.ca/home/nkanaya
home.starke	/afs/phys.uvic.ca/home/starke
vicepd	
home.agarwal	/afs/phys.uvic.ca/home/agarwal
home.atlas	/afs/phys.uvic.ca/atlas

3 Intended Usage of the phys.uvic.ca AFS cell, Spring 2003

With the previous usage of the AFS server, input data was fetched (read from the AFS server) fairly consistently throughout the execution of the job. The ATLAS executable scripts were fetched throughout the job, and the ATLAS output files were cached and then sent (written) to the AFS server as the local cache became full. The intended usage of AFS as a facilitator of executing ATLAS jobs on the grid in spring 2003 was only as the source for the actual ATLAS .zsh execution programs, but not for the input or output data files. With the proposed setup, seen in section 3.1, the ATLAS input and output files would be tarred, compressed and then transferred using multi-streamed, secure *Grid-FTP*. Thus, AFS would only be used for fetching the ATLAS .zsh executable, which is static.

3.1 ATLAS event simulation jobs - submitted using Globus and AFS, Spring 2003

The ATLAS event simulation jobs were submitted via globus, as follows:

1. Using globus-job-run, a Perl executable script is submitted to a remote grid host, which controls the following processes.
2. ATLAS input files are tarred and compressed on local host.
3. Using Grid-FTP, ATLAS input files are sent to the remote host.
4. ATLAS input files are untarred and uncompressed on remote host.
5. Remote host fetchs ATLAS .zsh event simulation scripts off the phys.uvic.ca AFS cell.
6. Execute the ATLAS job on the remote host.
7. As the job is executing, output data is written to remote host disk.
8. Upon completion, ATLAS output files are tarred and compressed on remote host.
9. Using Grid-FTP, ATLAS output files are sent back to the local host and then to the phys.uvic.ca AFS cell.

4 Proposed changes to the phys.uvic.ca AFS cell to Increase Efficiency

4.1 Clones and Replicated Volumes

All of the static .zsh ATLAS executables are stored in the home.atlas volume on the AFS server. When a volume is being used for read-only purposes, clones or replicated volumes can be made of the original read/write volume. In the OpenAFS Administration Manual it states that using read-only and replicated read-only volumes can facilitate a more efficient usage of AFS Server machines if the material in the volumes being replicated does not change very often. For a clear explanation of clones and replication volumes, as well as mount point traversals, please see Appendix A. Since the ATLAS execution files do not change unless a new ATLAS data Challenge is brought about, the home.atlas volume is a perfect candidate to create clones/replicated volumes. From Dan Vanderster's Report [12], it was noted that traffic bottlenecks occurred when the number of remote sites trying to contact the AFS server increased. This is one reason why creating clones or replication sites will help increase efficiency in the AFS cell. There will simply be more sites from which data can be accessed. However, in the OpenAFS Administration Manual it states that, as a rule, no more than one clone/replication volume can be created per AFS File Server machine. Thus, it would be an advantage to increase the number of AFS File Server machines in the phys.uvic.ca AFS cell [13].

The second reason why creating clones/replication volumes is thought to increase efficiency is due to how many remote procedure calls are made when data is read from a read/write volume compared to when it is read from a read-only volume. When a read/write volume is read by an AFS client, the cache manager caches the whole or as much as the whole volume being accessed as possible. Each time a file from within that volume is accessed, the AFS cache manager must contact the AFS server to see if any changes have been made to that file since it was cached on the AFS client machine. When a read-only volume is read by an AFS client, the cache manager also caches the whole or as much as the whole volume being accessed as possible. However, the cache manager does not need to check with the AFS server every time it accesses a different file because it knows that the whole volume is static. Thus, the CPU efficiency of the remote host

accessing data from the phys.uvic.ca AFS cell should increase dramatically while ATLAS is being executed over AFS, since the ATLAS event simulation application accesses thousands of files from a volume that doesn't change.

4.2 The submitted proposal

1. Add another machine to the phys.uvic.ca AFS cell to be used as a File Server Machine.
2. Make that machine a Database Server Machine as well.
3. Create a read-only clone on the same partition where home.atlas exists on basilisk.phys.uvic.ca.
4. Create a read-only replicated volume on the new File Server machine.

5 Accepted Setup of AFS, Spring 2003

The proposal submitted in Section 4.2 was accepted in February 2003 and the changes were put into place over a period of nearly 4 weeks. The new File Server/Database Server machine that was incorporated into the phys.uvic.ca AFS cell is called objy.phys.uvic.ca. Some of its hardware/software information can be seen in Table 3. The resulting changes in the volumes created and their mount points can be seen in Table 4 and Table 5. Details of the modifications can be seen in Section A.8. Note that the clone on basilisk.phys.uvic.ca and the replicated volume created on objy.phys.uvic.ca

Table 3: Objy.phys.uvic.ca Hardware/Software Information

MODEL	Pentium III	CPU	
RAM	256MB	Processor:	Single
OS	RedHat Linux 7.3 (valhalla)	Speed:	451MHz
DISKSPACE			
hda	12GB	Quantum Fireball	CX13.0A
hdb	97GB	WDC	WD1000BB-00CCB0
hdc	120GB	Maxtor	4G120J6
hdd	80GB	Samsung	SP8004H

have a .read-only suffix. It may have been observed by the reader in Table 2 that there were already some clones on basilisk.phys.uvic.ca, root.afs.read-only and root.cell.read-only, namely. These still remain in Table 4 and have also been added to Table 5. The reasons why these are necessary is explained in detail in Appendix A.

6 Observed Results and Conclusions due to changes made in the phys.uvic.ca AFS cell

Several tests that use AFS and Grid-FTP were done before the clone and replicated volume of home.atlas were made (the AFS setup described in Section 2.1). These tests were then repeated with the new clone and replicated volume (the AFS setup described in Section 5). The following sections describe the tests and their outcomes. It should be noted that all tests conducted were executing ATLAS 4.0.1 data challenge event simulation on the Globus 2.2.4 platform, using OpenAFS version 1.2.3. The following tests were performed following the procedure described in section 3.1.

6.1 Seperate Submission to Single Remote Hosts

The times recorded in Table 6 and 7 are from running the ATLAS 4.0.1 data challenge on two single remote hosts at seperate times. These times and CPU efficiencies[1] are then used as a base from which to compare all following tests. When they were ran they were the only remote hosts (each at seperate times) sourcing data from the phys.uvic.ca AFS cell. It can be observed in Figures 2 and 3 that the rates of data transfer from AFS peaked at about 25kBytes/s. Data transfer rates shown on all graphs are averages over five minutes. Since the four tests done in this section;

1. Job submitted to vision.triumf.ca without clones/replicated volumes
2. Job submitted to vision.triumf.ca with clones/replicated volumes
3. Job submitted to thuner-gw.phys.ualberta.ca without clones/replicated volumes

[1] CPU Efficiency is defined for this report to be (system time + user time)/(real time).

Table 4: Current Volume Mount Points on basilisk.phys.uvic.ca

VOLUME NAME	MOUNT POINT
vicepa	
public	/afs/phys.uvic.ca/public
root.afs	/afs
root.afs.readonly	/afs
root.cell	/afs/phys.uvic.ca
root.cell.readonly	/afs/phys.uvic.ca
vicepb	
home.babar.kanga	/afs/phys.uvic.ca/babar/kanga
home.gable	/afs/phys.uvic.ca/home/gable
home.gabriel	/afs/phys.uvic.ca/home/gabriel
home.gking	/afs/phys.uvic.ca/home/gking
home.jjallan	/afs/phys.uvic.ca/home/jjallan
home.manj	/afs/phys.uvic.ca/home/manj
home.sobie	/afs/phys.uvic.ca/home/sobie
home.wyvern	/afs/phys.uvic.ca/home/wyvern
home.zwiers	/afs/phys.uvic.ca/home/zwiers
vicepc	
home.babar	/afs/phys.uvic.ca/babar
home.babar.cern	/afs/phys.uvic.ca/babar/cern
home.babar.objectivity	/afs/phys.uvic.ca/babar/package/objectivity
home.babar.package	/afs/phys.uvic.ca/babar/package
home.babar.packages	/afs/phys.uvic.ca/babar/dist/packages
home.babar.releases	/afs/phys.uvic.ca/babar/dist/releases
home.babar.roguewave	/afs/phys.uvic.ca/babar/package/RogueWave
home.dvanders	/afs/phys.uvic.ca/home/dvanders
home.nkanaya	/afs/phys.uvic.ca/home/nkanaya
home.starke	/afs/phys.uvic.ca/home/starke
vicepd	
home.agarwal	/afs/phys.uvic.ca/home/agarwal
home.atlas	/afs/phys.uvic.ca/atlas
home.atlas.readonly	/afs/phys.uvic.ca/atlas

4. Job submitted to thuner-gw.phys.ualberta.ca with clones/replicated volumes

were all submitted seperately, there was never more than one remote grid resource trying to access the phys.uvic.ca AFS cell at once. Any increase in CPU efficiency must be attributed the different methods (that AFS uses for read/write and read-only volumes) that RPC's (call-backs) are used to check if the volumes are up-to-date, rather than the fact that more sites increase the accessibility of the data. See Section A.6 for a reminder on how call-backs work. The information in Table 6 shows that when only one job is being run, there is a for all practical matter, no increase or decrease in remote host CPU efficiency when clones/replicated volumes are added to the cell. However, in Table 7 there is a marked increase in remote host CPU

Table 5: Current Volume Mount Points on objy.phys.uvic.ca

VOLUME NAME	MOUNT POINT
vicepa	
root.afs.readonly	/afs
root.cell.readonly	/afs/phys.uvic.ca
vicepb	
home.atlas.write	/afs/phys.uvic.ca/atlas-write
vicepc	
home.atlas.readonly	/afs/phys.uvic.ca/atlas

Table 6: Job Submission times for **vision.triumf.ca** by itself

	Without Clone	With Clone
Grid-FTP (put):	4m41s	5m41s
Untar and Uncompress (remote):	5m4s	4m53s
Execution (real):	98m45s	99m3s
Execution (user):	86m43s	86m42s
Execution (system):	2m11s	2m9s
Tar and Compress (remote):	13m30s	13m5s
Grid-FTP (get):	1m12s	1m13s
Untar and Uncompress (local):	4m1s	3m57s
Total Job Run Time:	2h7m58s	2h8m20s
CPU Efficiency:	**90%**	**89.7%**

efficiency with the advent of the clone/replicated volume. In fact thuner-gw.phys.ualberta.ca is in Edmonton, AB (which is 1222 km from UVIC) and vision.triumf.ca is in Vancouver, BC (which is only 68 km from UVIC). Thus, perhaps due to the exellent transfer rates that can be achieved to TRIUMF, the increased efficiency of the phys.uvic.ca AFS cell cannot be observed, but the slower connection to the University of Alberta magnifies the changes in the phys.uvic.ca AFS cell, so that the difference is more noticable.

6.2 Simultaneous Submission on Two Remote Hosts

The times recorded in Table 8 and 9 are from running the ATLAS 4.0.1 data challenge on two single remote hosts, vision.triumf.ca and thuner-gw.phys.ualberta.ca, simultaneously. Again, what was observed in Section 6.1 can be observed in Tables 8 and 9. In Section 6.1 it was observed that due to the excellent connection from UVIC to TRIUMF (vision.triumf.ca), the increased efficiency at the phys.uvic.ca AFS cell, created by adding the clone and replicated volume, is unobservable when looking for a correlation with the remote host CPU efficiency. The same can be seen in Table 8, as the difference in the CPU efficiencies with and without clones in Table 8 is negligible. Similarly, it was observed in Section 6.1 that the distance between UVIC and the University of Alberta, and the latencies that the distance creates, help to emphasize the increased server efficiencies at the phys.uvic.ca AFS cell. This increased AFS efficiency can correspondingly be

Table 7: Job Submission times for **thuner-gw.phys.ualberta.ca** by itself

	Without Clone	With Clone
Grid-FTP (put):	7m14s	6m50s
Untar and Uncompress (remote):	8m17s	6m28s
Execution (real):	72m43s	66m31s
Execution (user):	47m48s	47m48s
Execution (system):	5m10s	5m27s
Tar and Compress (remote):	9m26s	8m43s
Grid-FTP (get):	2m40s	1m39s
Untar and Uncompress (local):	4m1s	4m0s
Total Job Run Time:	1h44m40s	1h34m33s
CPU Efficiency:	**72.8%**	**80.0%**

Figure 2: **Single Hosts Without Clones - Seperate Submission**
The two peaks in this graph, at 00:15 and 11:30 (ignore peak at 17:30), show the ATLAS .zsh executable being grabbed by thuner-gw.phys.ualberta and vision.triumf.ca, respectively, from the phys.uvic.ca AFS cell.

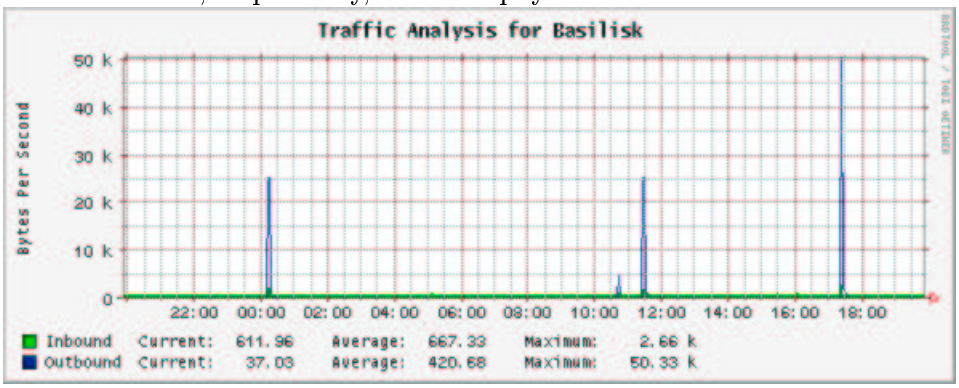
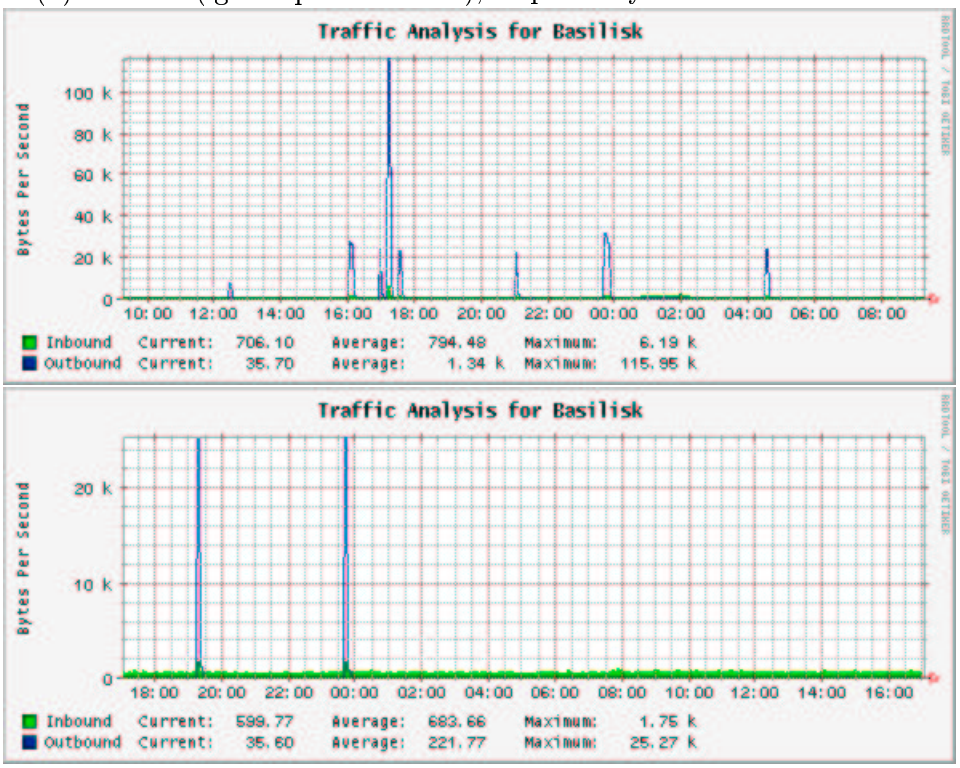


Table 8: Job Submission times for **vision.triumf.ca** at the same time as thuner-gw.phys.ualberta.ca

	Without Clone	With Clone
Grid-FTP (put):	10m17s	10m42s
Untar and Uncompress (remote):	4m57s	4m55s
Execution (real):	99m15s	98m40s
Execution (user):	86m46s	86m44s
Execution (system):	2m10s	2m11s
Tar and Compress (remote):	13m16s	13m11s
Grid-FTP (get):	1m13s	1m13s
Untar and Uncompress (local):	4m1s	4m1s
Total Job Run Time:	2h13m28s	2h13m25s
CPU Efficiency:	**89.6%**	**90.1%**

Figure 3: **Single Hosts With Clones - Seperate Submission**
The two peaks in these graphs show the ATLAS .zsh executable being grabbed by thuner-gw.phys.ualberta from basilisk.phys.uvic.ca in (a) at 04:30 (ignore all other peaks) and vision.triumf.ca from basilisk.phys.uvic.ca in (b) at 23:45 (ignore peak at 19:15), respectively.



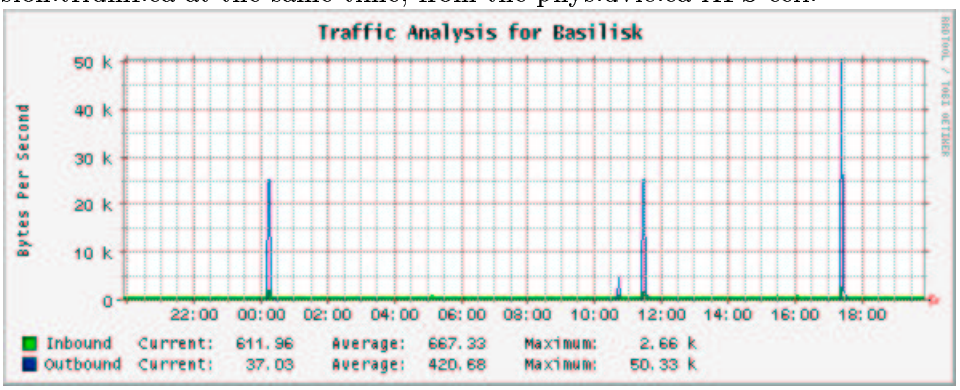
seen through the increase in the CPU efficiency at the remote site, thuner-gw.phys.ualberta.ca.

Another observation in Section 6.1 was that the increase in phys.uvic.ca AFS cell efficiency could not be because there are data sites (clones/replication volume) as there was only one host ever accessing the data. In this section, however, there are two remote hosts accessing the data off of the phys.uvic.ca AFS cell. Still, when looking at Figure 4, which shows simultaneous submission to vision.triumf.ca and thuner-gw.phys.ualberta.ca without clones, it can be seen when compared to Figure 2, which shows seperate submission to vision.triumf.ca and thuner-gw.phys.ualberta.ca without clones, that AFS has not been saturated. The transfer rates are simply doubled and twice up much data moves twice as fast. What is even stranger is that in the simultaneous submission to vision.triumf.ca and thuner-gw.phys.ualberta.ca, as seen in Figure 5, only basilisk.phys.uvic.ca is accessed. It was expected that both AFS Server machines would be accessed. The reason for this is unknown. What is known, however, is that the individual transfer rates are not increased due to the increased accessibility of the data on the phys.uvic.ca AFS cell. Instead, the increased CPU efficiencies seen in Table 9 must be attributed to the smarter call-back method used by AFS when accessing data from a read-only volume.

Table 9: Job Submission times for **thuner-gw.phys.ualberta.ca** at the same time as vision.triumf.ca

	Without Clone	With Clone
Grid-FTP (put):	10m26s	10m27s
Untar and Uncompress (remote):	6m47s	6m53s
Execution (real):	102m48s	66m29s
Execution (user):	49m6s	47m48s
Execution (system):	9m49s	5m23s
Tar and Compress (remote):	19m50s	8m17s
Grid-FTP (get):	3m29s	2m37s
Untar and Uncompress (local):	3m58s	3m58s
Total Job Run Time:	2h25m52s	1h39m3s
CPU Efficiency:	**57.3%**	**79.5%**

Figure 4: **Single Hosts Without Clones - Simultaneous Submission** The peak in this graph, at 17:20 (ignore peaks at 00:15 and 11:30), is the ATLAS .zsh executable being grabbed by thuner-gw.phys.ualberta and vision.triumf.ca at the same time, from the phys.uvic.ca AFS cell.



6.3 Simultaneous Submission to Multiple Remote Hosts

Since up to this point, no increased efficiency has been seen due to the increased accessibility, it was decided to try submitting the ATLAS 4.0.1 data challenge to a multitude of remote grid resources simultaneously and to observe the results on the CPU efficiencies of the base remote sites, vision.triumf.ca and thuner-gw.phys.ualberta.ca. Again, this test was done without and then with the clones/replicated volumes. In both cases, the ATLAS jobs were submitted to six sites at the same time. The observed difference on vision.triumf.ca can be seen in Table 10 and on thuner-gw.phys.ualberta.ca in Table 11. For both of the tests, with and without clones, the ATLAS 4.0.1 data challenge was submitted to six remote machines simultaneously. Although, it can be seen from the data in Tables 10 and 11 that there was no marked increase in the base machines (vision.triumf.ca and thuner-gw.phys.ualberta.ca) CPU efficiencies with the advent of the clones, what is remarkable is that, of the six jobs submitted to remote machines, six finished with the clones but only four finished without the clones. The reason that they did not finish is because the AFS connection timed out. Those remote hosts were unable to establish a connection with AFS, since the server was too busy. So, even though the individual CPU efficiencies did not improve, it can be seen that the accessibility of the data due to the extra volume sites (clones/replicated volumes) does increase the efficiency of the phys.uvic.ca

Figure 5: **Single Hosts With Clones - Simultaneous Submission** The peak or lack of peak in these graphs, at 19:15 (ignore peak at 13:50 on (a) and peak at 22:45on (b)), is the ATLAS .zsh executable being grabbed by thuner-gw.phys.ualberta and vision.triumf.ca at the same time, from the phys.uvic.ca AFS cell. (a) objy.phys.uvic.ca, (b) basilisk.phys.uvic.ca. Note that data is only accessed from basilisk.phys.uvic.ca

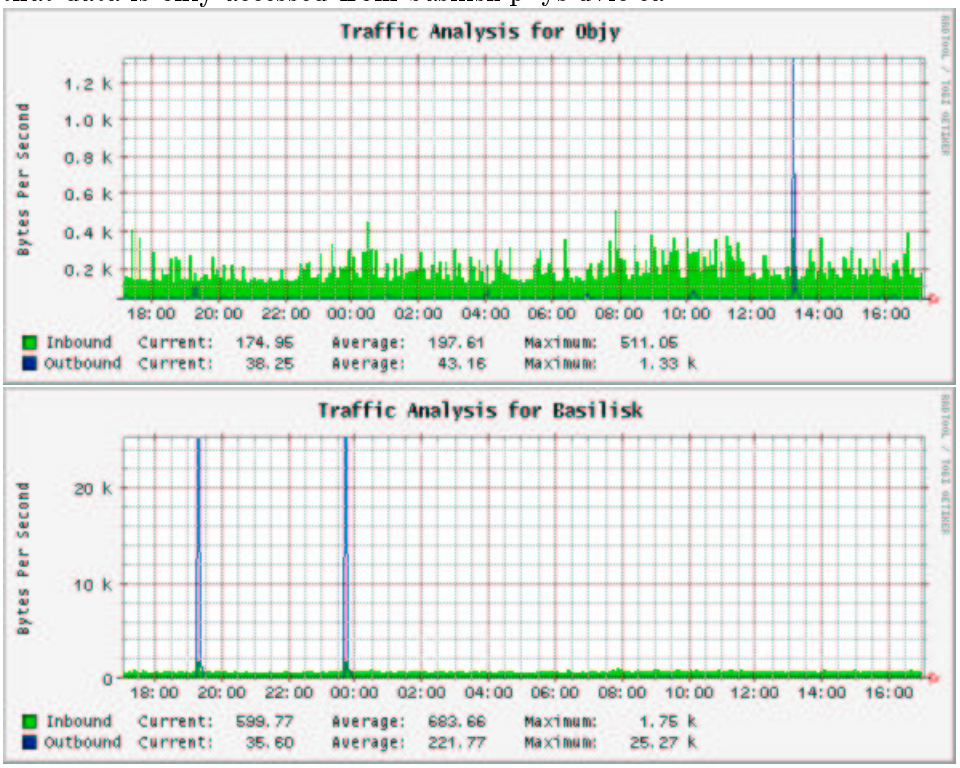


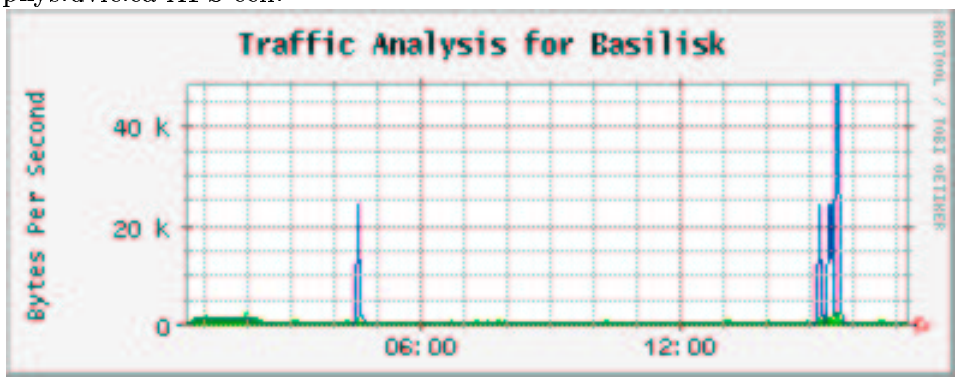
Table 10: Job Submission times for **vision.triumf.ca**, when submitted to a total of six grid resource sites simultaneously.

	Without Clone	With Clone
Grid-FTP (put):	27m6s	44m20s
Untar and Uncompress (remote):	5m3s	5m6s
Execution (real):	98m50s	98m11s
Execution (user):	86m41s	86m41s
Execution (system):	2m10s	2m10s
Tar and Compress (remote):	13m16s	13m16s
Grid-FTP (get):	1m21s	1m48s
Untar and Uncompress (local):	4m27s	4m47s
Total Job Run Time:	2h30m41s	2h48m27s
CPU Efficiency:	**89.9%**	**90.5%**

Table 11: Job Submission times for **thuner-gw.phys.ualberta.ca**, when submitted to a total of six grid resource sites simultaneously.

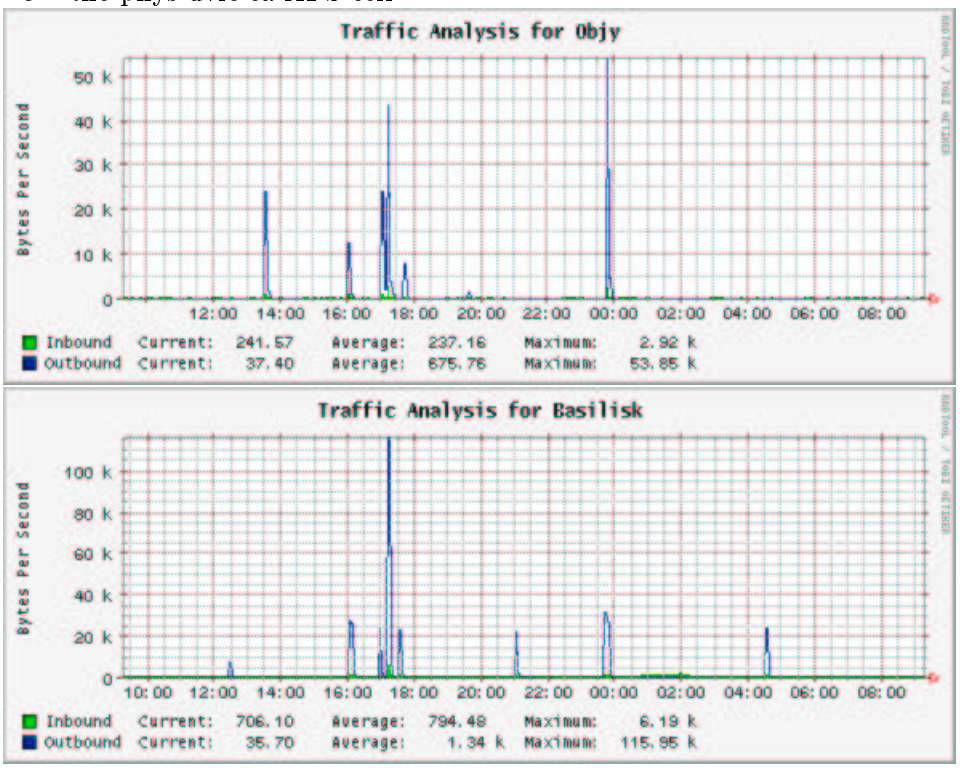
	Without Clone	With Clone
Grid-FTP (put):	26m16s	43m58s
Untar and Uncompress (remote):	6m34s	7m14s
Execution (real):	66m22s	70m30s
Execution (user):	47m47s	48m11s
Execution (system):	5m26s	6m16s
Tar and Compress (remote):	8m24s	8m24s
Grid-FTP (get):	2m23s	3m37s
Untar and Uncompress (local):	4m29s	4m48s
Total Job Run Time:	1h55m17s	2h19m36s
CPU Efficiency:	**80.1%**	**77.2%**

Figure 6: **Multiple Hosts Without Clones - Simultaneous Submission**
The peaks in this graph, at 15:40 (ignore peak at 4:30), is the ATLAS .zsh executable being grabbed by the remote hosts at the same time, from the phys.uvic.ca AFS cell.



AFS cell. Figure 6 shows the traffic on the phys.uvic.ca AFS cell when the four jobs are being run simultaneously without clones. Of course, all of the traffic is on Basilisk.phys.uvic.ca because there is no clone to be accessed off of objy.phys.uvic.ca. The transfer rates peak at about 50 KBytes/s. In Figure 7 the traffic on the phys.uvic.ca AFS cell can be seen when the six jobs are being run simultaneously with clones. The remote sites access the data from volume sites on both of the phys.uvic.ca AFS servers. Transfer rates peak at about 60 KBytes/s on Objy.phys.uvic.ca and at about 33 KBytes/s on Basilisk.phys.uvic.ca. It would seem accurate to say that even submitting jobs to six remote hosts simultaneously, does not tax the phys.uvic.ca AFS system when it has clones. However, without clones there are too many remote AFS clients trying to contact and connect with the phys.uvic.ca AFS cell. Thus, some of the connections time out and the completion of the job is aborted. This seems conclusive in proving that adding the clones/replicated volumes to the phys.uvic.ca AFS cell has increased the efficiency with which the system is accessed.

Figure 7: **Multiple Hosts With Clones - Simultaneous Submission** The peak in these graphs, at 23:45 (ignore all other peaks), is the ATLAS .zsh executable being grabbed by the six remote hosts at the same time, from the phys.uvic.ca AFS cell.



7 Conclusion

The tests that were conducted are conclusive in ascertaining that the improvements increased the phys.uvic.ca AFS cell efficiency, making it faster for remote grid resources to access data from it. The test results described in Sections 6.1 and 6.2 show that the different method used for call-backs by AFS when there are clones/replicated volumes compared to no clones increases the phys.uvic.ca AFS cell. The test results described in Section 6.3 shows that by adding more physical sites (clones and replicated volumes) the data becomes more accessible and the phys.uvic.ca AFS cell can handle being simultaneously accessed by more remote machines then without clones/replicated volumes.

8 Recommendations

Although the tests that were conducted are conclusive in ascertaining that the improvements increased the phys.uvic.ca AFS cell efficiency, it would be a good idea to conduct more tests along the same lines, but continuously, to prove that AFS can handle continuous access over extended periods of time. Another recommendation would be to redo the tests with a different connection, because the connection hub to the phys.uvic.ca AFS cell is a suspect as a cause for slowing down transfer rates. Finally, if this setup were to be committed for production use, it would be advised to invest in newer machines as the phys.uvic.ca AFS cell Server machines.

References

[1] "Grid Computing: The Basics" [Webpage] Available at: http://www.grid.org/about/gc/

[2] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International J. Supercomputer Applications*, Vol.15 (3), 2001.

[3] I. Foster, "The Grid: Computing without Bounds", *Scientific American*, Vol.288 (4) pp.78-85, Apr.2003.

[4] "Canada's Research and Innovation Network" [Webpage] Available at: http://www.canarie.ca/

[5] "Canadian High Performance Computing Collaboratory" [Webpage] Available at: http://www.C3.ca/

[6] "The National Research Council" [Webpage] Available at: http://www.nrc-cnrc.gc.ca/

[7] Gridmaster, "Grid Canada" [Webpage] Available at: http://www.gridcanada.ca/about.html

[8] Gridmaster, "UVIC Grid Testbed" [Webpage] Available at: http://grid.phys.uvic.ca

[9] M. Barnett, "The ATLAS Experiment Homepage" [Webpage] Available at: http://pdg.lbl.gov/atlas/atlas.html

[10] "CERN" [Webpage] Available at: http://www.http://public.web.cern.ch/public/

[11] "The Condor Project - Homepage" [Webpage] Available at: http://www.cs.wisc.edu/condor/

[12] D.Vanderster, "Grid Computing Using Particle Physics Applications", UVIC Computer Engineering Co-op Report, pp.15-17, 2002.

[13] Webmaster, "OpenAFS" [Webpage] (Mar 2003) Available at: http://www.openafs.org

[14] M. I. Williams, C. Hee, B. Barrera, T. Glanzman, "The BaBar Homepage", [Webpage] (Mar 2003) Available at: http://www.slac.stanford.edu/BFROOT/

A AFS: A Distributed File System

The following is excerpted from the Administration Guide off of the OpenAFS website. For more information please see [13].

AFS is a distributed file system that enables users to share and access all of the files stored in a network of computers as easily as they access the files stored on their local machines. The file system is called distributed for this exact reason: files can reside on many different machines (be distributed across them), but are available to users on every machine.

A.1 Servers and Clients

In fact, AFS stores files on a subset of the machines in a network, called file server machines. File server machines provide file storage and delivery service, along with other specialized services, to the other subset of machines in the network, the client machines. These machines are called clients because they make use of the servers' services while doing their own work. In a standard AFS configuration, clients provide computational power, access to the files in AFS and other "general purpose" tools to the users seated at their consoles. There are generally many more client workstations than file server machines.

AFS file server machines run a number of server processes, so called because each provides a distinct specialized service: one handles file requests, another tracks file location, a third manages security, and so on. To avoid confusion, AFS documentation always refers to server machines and server processes, not simply to servers. For a more detailed description of the server processes, see AFS Server Processes and the Cache Manager.

A.2 Cells

A cell is an administratively independent site running AFS. As a cell's system administrator, you make many decisions about configuring and maintaining your cell in the way that best serves its users, without having to consult the administrators in other cells. For example, you determine how many clients and servers to have, where to put files, and how to allocate client machines to users.

A.3 Transparent Access and the Uniform Namespace

Although your AFS cell is administratively independent, you probably want to organize the local collection of files (your filespace or tree) so that users from other cells can also access the information in it. AFS enables cells to combine their local filespaces into a global filespace, and does so in such a way that file access is transparent–users do not need to know anything about a file's location in order to access it. All they need to know is the pathname of the file, which looks the same in every cell. Thus every user at every machine sees the collection of files in the same way, meaning that AFS provides a uniform namespace to its users.

A.4 Volumes

AFS groups files into volumes, making it possible to distribute files across many machines and yet maintain a uniform namespace. A volume is a unit of disk space that functions like a container for a set of related files, keeping them all together on one partition. Volumes can vary in size, but are (by definition) smaller than a partition.

Volumes are important to system administrators and users for several reasons. Their small size makes them easy to move from one partition to another, or even between machines. The system administrator can maintain maximum efficiency by moving volumes to keep the load balanced evenly. In addition, volumes correspond to directories in the filespace–most cells store the contents of each user home directory in a separate volume. Thus the complete contents of the directory move together when the volume moves, making it easy for AFS to keep track of where a file is at a certain time. Volume moves are recorded automatically, so users do not have to keep track of file locations.

A.5 Mount Points

The directory that corresponds to a volume is called its root directory, and the mechanism that associates the directory and volume is called a mount point. A mount point is similar to a symbolic link in the file tree that specifies which volume contains the files kept in a directory. A mount point is not an actual symbolic link; its internal structure is different. A volume is said to be mounted at the point in the file tree where there is a mount point

pointing to the volume. A volume's contents are not visible or accessible unless it is mounted.

A.5.1 The Three types of Mount Points

AFS uses three types of mount points, each appropriate for a different purpose because of how the Cache Manager handles them.

- When the Cache Manager crosses a regular mount point, it obeys all three of the mount point traversal rules previously described.

  AFS performs best when the vast majority of mount points in the filespace are regular, because the mount point traversal rules promote the most efficient use of both replicated and nonreplicated volumes. Because there are likely to be multiple read-only copies of a replicated volume, it makes sense for the Cache Manager to access one of them rather than the single read/write version, and the second rule leads it to do so. If a volume is not replicated, the third rule means that the Cache Manager still accesses the read/write volume when that is the only type available. In other words, a regular mount point does not force the Cache Manager always to access read-only volumes (it is explicitly not a "read-only mount point").

  Note: To enable the Cache Manager to access the read-only version of a replicated volume named by a regular mount point, all volumes that are mounted above it in the pathname must also be replicated. That is the only way the Cache Manager can stay on a read-only path to the target volume.

- When the Cache Manager crosses a read/write mount point, it attempts to access only the volume version named in the mount point. If the volume name is the base (read/write) form, without a .readonly or .backup extension, the Cache Manager accesses the read/write version of the volume, even if it is replicated. In other words, the Cache Manager disregards the second mount point traversal rule when crossing a read/write mount point: it switches to the read/write path through the filespace.

  It is conventional to create only one read/write mount point in a cell's filespace, using it to mount the cell's root.cell volume just below the AFS filespace root (by convention, /afs/.cellname). As indicated, it

is conventional to place a period at the start of the read/write mount point's name (for example, /afs/.abc.com). The period distinguishes the read/write mount point from the regular mount point for the root.cell volume at the same level. This is the only case in which it is conventional to create two mount points for the same volume. A desirable side effect of this naming convention for this read/write mount point is that it does not appear in the output of the UNIX ls command unless the -a flag is included, essentially hiding it from regular users who have no use for it.

The existence of a single read/write mount point at this point in the filespace provides access to the read/write version of every volume when necessary, because it puts the Cache Manager on a read/write path right at the top of the filespace. At the same time, the regular mount point for the root.cell volume puts the Cache Manager on a read-only path most of the time.

Using a read/write mount point for a read-only or backup volume is acceptable, but unnecessary. The first rule of mount point traversal already specifies that the Cache Manager accesses them if the volume name in a regular mount point has a .readonly or .backup extension.

- When the Cache Manager crosses a cellular mount point, it accesses the indicated volume in the specified cell, which is normally a foreign cell. (If the mount point does not name a cell along with the volume, the Cache Manager accesses the volume in the cell where the mount point resides.) When crossing a regular cellular mount point, the Cache Manager disregards the third mount point traversal rule. Instead, it accesses a read-only version of the volume if it is replicated, even if the volume that houses the mount point is read/write.

  It is inappropriate to circumvent this behavior by creating a read/write cellular mount point, because traversing the read/write path imposes an unfair load on the foreign cell's file server machines. The File Server must issue a callback for each file fetched from the read/write volume, rather than single callback required for a read-only volume. In any case, only a cell's own administrators generally need to access the read/write versions of replicated volumes.

  It is conventional to create cellular mount points only at the second level in a cell's filespace, using them to mount foreign cells' root.cell volumes just below the AFS filespace root (by convention, at /afs/foreign_cellname). The mount point enables local users to access the foreign cell's filespace,

assuming they have the necessary permissions on the ACL of the volume's root directory and that there is an entry for the foreign cell in each local client machine's /usr/vice/etc/CellServDB file, as described in Maintaining Knowledge of Database Server Machines.

Creating cellular mount points at other levels in the filespace and mounting foreign volumes other than the root.cell volume is not generally appropriate. It can be confusing to users if the Cache Manager switches between cells at various points in a pathname.

A.5.2 Rules of Mount Point Traversals

The Cache Manager observes three basic rules as it traverses the AFS filespace and encounters mount points:

1. Access Backup and Read-only Volumes When Specified

   When the Cache Manager encounters a mount point that specifies a volume with either a .readonly or a .backup extension, it accesses that type of volume only. If a mount point does not have either a .backup or .readonly extension, the Cache Manager uses Rules 2 and 3.

   For example, the Cache Manager never accesses the read/write version of a volume if the mount point names the backup version. If the specified version is inaccessible, the Cache Manager reports an error.

2. Follow the Read-only Path When Possible

   If a mount point resides in a read-only volume and the volume that it references is replicated, the Cache Manager attempts to access a read-only copy of the volume; if the referenced volume is not replicated, the Cache Manager accesses the read/write copy. The Cache Manager is thus said to prefer a read-only path through the filespace, accessing read-only volumes when they are available.

   The Cache Manager starts on the read-only path in the first place because it always accesses a read-only copy of the root.afs volume if it exists; the volume is mounted at the root of a cell's AFS filespace (named /afs by convention). That is, if the root.afs volume is replicated, the Cache Manager attempts to access a read-only copy of it rather than the read/write copy. This rule then keeps the Cache Manager on a read-only path as long as each successive volume is replicated. The implication is that both the root.afs and root.cell volumes

must be replicated for the Cache Manager to access replicated volumes mounted below them in the AFS filespace. The volumes are conventionally mounted at the /afs and /afs/cellname directories, respectively.

3. Once on a Read/write Path, Stay There

   If a mount point resides in a read/write volume and the volume name does not have a .readonly or a .backup extension, the Cache Manager attempts to access only the a read/write version of the volume. The access attempt fails with an error if the read/write version is inaccessible, even if a read-only version is accessible. In this situation the Cache Manager is said to be on a read/write path and cannot switch back to the read-only path unless mount point explicitly names a volume with a .readonly extension. (Cellular mount points are an important exception to this rule, as explained in the following discussion.

A.6 Efficiency Boosters: Replication and Caching

AFS incorporates special features on server machines and client machines that help make it efficient and reliable.

On server machines, AFS enables administrators to replicate commonly-used volumes, such as those containing binaries for popular programs. Replication means putting an identical read-only copy (It's called a clone when the replicated copy is put on the same partition as the original read/write volume. Clones are not identical copies but only a database header and an index of the vnode encompaased by the read/write volume.) of a volume on more than one file server machine. The failure of one file server machine housing the volume does not interrupt users' work, because the volume's contents are still available from other machines. Replication also means that one machine does not become overburdened with requests for files from a popular volume.

On client machines, AFS uses caching to improve efficiency. When a user on a client workstation requests a file, the Cache Manager on the client sends a request for the data to the File Server process running on the proper file server machine. The user does not need to know which machine this is; the Cache Manager determines file location automatically. The Cache Manager receives the file from the File Server process and puts it into the cache, an area of the client machine's local disk or memory dedicated to temporary

file storage. Caching improves efficiency because the client does not need to send a request across the network every time the user wants the same file. Network traffic is minimized, and subsequent access to the file is especially fast because the file is stored locally. AFS has a way of ensuring that the cached file stays up-to-date, called a callback.

A.7 Security: Mutual Authentication and Access Control Lists

Even in a cell where file sharing is especially frequent and widespread, it is not desirable that every user have equal access to every file. One way AFS provides adequate security is by requiring that servers and clients prove their identities to one another before they exchange information. This procedure, called mutual authentication, requires that both server and client demonstrate knowledge of a "shared secret" (like a password) known only to the two of them. Mutual authentication guarantees that servers provide information only to authorized clients and that clients receive information only from legitimate servers.

Users themselves control another aspect of AFS security, by determining who has access to the directories they own. For any directory a user owns, he or she can build an access control list (ACL) that grants or denies access to the contents of the directory. An access control list pairs specific users with specific types of access privileges. There are seven separate permissions and up to twenty different people or groups of people can appear on an access control list.

A.8 Details of Changes made to the phys.uvic.ca AFS cell

A second machine, objy.phys.uvic.ca, has been added to the phys.uvic.ca AFS cell. Objy.phys.uvic.ca now runs as a Database Server and a File Server Machine. First, read-only replication volumes were made of the root.afs and root.cell volumes onto partition vicepa on objy.phys.uvic.ca. Secondly, a new read/write volume, home.atlas.write was created on partition vicepb on objy.phys.uvic.ca and mounted at /afs/phys.uvic.ca/atlas-write/. This is where all output from any Atlas jobs should be sent to. (Or any directory that you wish to send it to, within that directory. New directories can be made by anyone in the atlas group.) Also any executable scripts that are being changed on a daily basis should be stored at /afs/phys.uvic.ca/atlas-

write/. All unnecessary (scripts/files/output logs/output data/ that change often) files that were located within /afs/phys.uvic.ca/atlas have been moved to /afs/phys.uvic.ca/atlas-write.

A new hard-drive was added to objy.phys.uvic.ca and a new partition /vicepc was created on it. A read-only replication of the home.atlas volume was created on /vicepc of objy.phys.uvic.ca and a read-only clone of home.atlas was created on /vicepd on basilisk.phys.uvic.ca. Write permission have been removed for regular users (non-administrative) on the home.atlas volume (which is mounted at /afs/phys.uvic.ca/atlas.

B The Globus Toolkit

The Globus Project is a research and development project focused on enabling the application of Grid concepts to scientific and engineering computing. (See below for an explanation of the Grid.)

- Groups around the world are using the Globus Toolkit to build Grids and to develop Grid applications.
- Globus Project research targets technical challenges that arise from these activities. Typical research areas include resource management, data management and access, application development environments, information services, and security.
- Globus Project software development has resulted in the Globus Toolkit, a set of services and software libraries to support Grids and Grid applications. The Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability.

B.1 User Commands

Globus jobs can be submitted directly if written in RSL (Resource Specification Language) but for the normal user, the Globus Team has provided easier to use wrappers. The most commonly used ones are described below. For all of them except the first to work, you must have grid-proxy on the local machine.

- *grid-proxy-init* This command creates a proxy for the unix user. It prompts you to enter your grid-proxy password that matches your Public Key Interface RSA Certificate/Key pair, and then proceeds with authentication.

- *grid-proxy-info* This command gives you information about your grid-proxy. It tells you the subject name and the amount time remaining on the proxy that is held by the unix user.

- *grid-proxy-destroy* This command destroys the proxy of the unix user.

- *globus-job-run hostname:portnumber executable* This command submits a job to a remote grid resource. It takes the arguments *hostname:portnumber* and *executable*. The executable can be from the remote host (default) or it can be stagged from the local machine by including *-s* before the *executable*. The results of the executable are by default sent to standard out.

- *globus-job-submit hostname:portnumber executable* This command is the same as *globus-job-run* except that it submits the job in batch mode. It takes the same arguments as *globus-job-run*. The results of the executable are by default saved to the unix users home directory. A string of the *jobid* is output to standard output.

- *globus-job-status jobid* This command takes the argument of a valid *jobid* and returns to standard out one of the four statuses: PENDING, ACTIVE, DONE or FAILED.

- *globus-job-get-output jobid* This command takes the argument of a valid *jobid* and returns to standard out the results of the *executable*.

- *globus-url-copy* [*options*] *source_url destination_url* This command does a Grid-FTP (secure file transfer) from the first location to the second location. If the location is local it must be defined as file:*/full_path/filename* and if it is remote it must be defined as gsiftp://*remote_hostname/full_path/filename*.

Third party transfers are allowed as long as you have a grid-proxy on the source machine. There are options to change the number of parallel streams, block size and buffer size.