Using AFS As A Distributed File System For Computational And Data Grids In High Energy Physics

MANCHESTER
1824

The University of Manchester

Using AFS As A Distributed File System For Computational And Data Grids In High Energy Physics

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Science and Engineering.

2005

Michael Angus Scott Jones

High Energy Particle Physics Group

Department of Physics and Astronomy

Contents

List of Figures

List of Tables

Abstract

The use of the distributed file system, AFS, as a solution to the "input/output sandbox" problem in grid computing is studied.

A computational grid middleware, primarily to accommodate the environment of the BaBar Computing Model, has been designed, written and is presented. A summary of the existing grid middleware and resources is discussed.

A number of benchmarks (one written for this thesis) are used to test the performance of the AFS over the wide area network and grid environment. The performance of the AFS is also tested using a straightforward BaBar Analysis code on real data.

Secure web-based and command-line interfaces created to monitor job submission and grid fabric are presented.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

About The Author

The author graduated in 1996 with a second class Bachelors Degree in Physics with Technological Physics from the University of Manchester. Following this he joined the High Energy Particle Physics group where he submitted the thesis "Measurement Of Depletion Voltage And Leakage Current For Silicon Strip Detectors", gaining a Masters Degree in Experimental Particle Physics in 1997.

He then went on to study δ-ray emission in the Forward Tracker at the H1 experiment at DESY with Liverpool University and later Hadronisation of Open Charm in Deep Inelastic Scattering through the channel $D^{\star\pm} \rightarrow D^{\circ}\pi^{\pm}_{Slow} \rightarrow K^{\mp}\pi^{\pm}\pi^{\pm}_{Slow}$. During this time he was jointly responsible for the operation of the forward tracking detectors. He developed an on-line monitoring system for H1 to collect and present histories of various detector conditions, beam conditions and collimator positions to help determine what was responsible for the poor conditions observed in the H1 detector during the 1998-9 electron proton runs.

Upon his return to the UK he became interested in emerging grid computing technologies and, in September 2001, joined the University of Manchester as a Research Associate in eScience with the local High Performance Computing department. Maintaining ties with the Particle Physics group at Manchester, he started a part time PhD in April 2002, exploring the use of AFS in BaBarGrid...

Preface

This thesis is of a multi-disciplinary nature. It is based on research in the fairly new and dynamic field of electronic science and research (eScience and eResearch). Whereas, the applications targeted for study and proof-of-concept were in the field of particle physics.

Grid computing and eScience technologies exist within a minefield of acronyms, abbreviation and jargon. In this thesis, where appropriate, I have written acronyms out in full and indicated (in parenthesis) the common form of abbreviation upon their first appearance. A Glossary/Abbreviations chapter has also been provided at the end of the main thesis.

The use of *an italic typeface* has been used where words may have specific meaning in the context of the technologies being discussed. The use of `a typewriter typeface` has been used to indicate computer system commands, file paths and Uniform Resource Locators (URLs).

Acknowledgements

I'm not a chatty writer so I'll get down to it straight away. In semi-no-particular order:

Thank you Katy for being, among other people, Katy.

Thank you Mum.

Thank you Phoebe.

Thank you Dad.

Thank you Lee, Rich (& Lisa), Rich, Saul, Andy, Marcus for remaining.

Thank you Jon, Mark, Matt and Stephen.

Thank you Alessandra. Andrew and Roger.

Thank you Manchester HEP folk for hospitality.

Thank you Manchester Computing folk for the job and the artificial lighting.

Thank you GridPP for the ability to collaborate.

Thank you Glen, Dave, Jon and Dan for having been in Hamburg.

Thank you Tim, Steve, Girish, Michael, Norma, Christopher, and National Express for making Liverpool bearable.

Thank you Paul for your support.

Thank you Iain for the sofa I wrote most of this on.

Thank you Platt.

Thank you Jess for staying up all night with me and this thesis.

$\sim$ Thank God $\sim$

Hello, my name is Inigo Montoya; you killed my father; prepare to die.

The Princess Bride, William Goldman

Chapter 1

Introduction

The purpose of the thesis is to study the fitness of the AFS [1] (formerly the Andrew File System) as a solution to the so called "input/output sandbox problem"[1]. This problem occurs when an executable is run on a remote system in a foreign environment. The files that the executable may expect to be able to read are not necessarily accessible or locally available in this environment. Also any output files produced will be stored somewhere in the remote filesystem and require special handling to retrieve.

A similar scenario exists with local batch systems. However, solutions in this environment are much less complex because the network is not generally so hostile. The popular approach in this environment is to mount a filesystem such as the Network File System (NFS) or the Storage Area Network (SAN). The success of the NFS/SAN approach in a local cluster provides the motivation for examining distributed file systems over the wide area network, internet and grid.

The grid environment for the HEP community has been chosen and is based on the Globus Toolkit® version 2 [2]. In this environment, is it feasible to use AFS as

[1]The input/output sandbox should not be confused with the concept of the execution sandbox. It is not, for example, the Java Sandbox, which is designed to limit the reach of system calls made by a running java applet.

a distributed file system? Does it perform adequately? Is it possible to effect the use of AFS in a seamless way so as to allow the scientists to expand their analyses into this grid environment with minimal impact?

The rest of this chapter describes the phenomenology of *the Grid*, how this fits in with the requirements of computing models in particle physics and outlines the structure of the rest of the thesis.

1.1 The Grid, An Historical Perspective

The Grid has been defined a number of ways by different people, groups and organisations. Here are a handful of those definitions.

> A grid must be able to *Find and share Data*, *Find and share Applications* and *Share Computing resources* [3].

> A grid is a software framework providing layers of services to access and manage distributed hardware and software resources. NASA's IPG (Information Power Grid) effort is an example of a grid [4].

> Grid: Widely distributed network of high-performance computers, stored data, instruments, and collaboration environments shared across institutional boundaries [5].

> The grid is the web on steroids [6].

> [The data-grid is] Napster for Scientists. [7]

> [The Grid] is distributed computing re-badged [8].

> Grid computing is non-trivial distributed computing across multiple administrative domains. [9]

> [The Grid] Provides flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resource. [10]

> [Of the "Grid Problem"] enables communities ("virtual organizations") to share geographically distributed resources as they pursue common goals – assuming the absence of central location, central control, omniscience, existing trust relationships. [11]

Therefore a grid is distributed but more so than the conventional clusters or Beowulf systems; it is spread across many sites and is subject to policies and procedures established at those sites. It must be able to deal with these complexities.

A grid should avoid central control. Where a grid would depend on a centrally controlled system, it introduces a single point of failure, presents a target for Denial of Service (DoS) attacks and offers an attractive target to hackers who seek to subvert its authority by acquiring its network identity or credential repository.

A grid must be maintainable in a scalable way. There should be no need for it to be overseen. It is easy to run a distributed application if a team of scientists are continually monitoring and restarting processes, but in that case the science would be in the operation of that grid and not in the scientific application, which would defeat the purpose of that application being grid-enabled.

The following subsections describe how grids have evolved over the last 15 years or so. We follow the evolution of disparate services (section 1.1.1) into collections (1.1.2) and finally uniformly accessible services (1.1.2).

1.1.1 Metacomputing

Around the end of the 1980s the concept of a new kind of computing paradigm was begining to emerge. A number of showcase projects started to try and utilise compute resources available at a number of supercomputer sites across America and indeed the world. This was the birth of Metacomputing.

One of these projects has a distinct synergy with this thesis. The I-WAY [12] project, which was demonstrated at the Supercomputing conference in 1995, is the

forerunner to the Globus Toolkit®. Furthermore, for part of its data transfer mechanisms it used a cell in the AFS. However, in this project, the use of AFS was limited as many of the systems involved did not have an AFS client installed. 'I-POP' servers were set up to act as a gateway between the remote systems and relay data to and from them and the AFS.

1.1.2 The Grid

Metacomputing was concerned more with the task of bridging the internet gaps between supercomputers than creating the ubiquity that the Grid addresses today. The resulting security issues, although acknowledged, had not been tackled.

Metacomputing evolved into grid computing when a simplistic analogy between the electricity grid and metacomputing was drawn. The idea was that one could plug their computer into a socket in the wall and draw on the teraFLOPs of processing power, access petabytes of data or even provide some of their resource back to the pool of recources [13, 14]. This is referred to as *the Grid*.

It is not the author's intention to convey the impression that *the Grid* exists or is indeed realisable in the near future. Rather, *a grid* refers to a subset of properties that *the Grid* would have.

To create *a grid* one needs a middleware layer between operating system software and client-side tools to expose the functionality of the operating system in a uniform way. A number of middleware technologies have emerged since the mid-nineties:

Globus Toolkit®

Globus Toolkit®version 1 (and later 2) is a collection of software implementing many different protocols wrapped in a PKI[1]-like security infrastructure. It provides

[1]Public Key Infrastructure [15]

functionality in three main categories: Resource Management, the ability to submit jobs to abstract queues on remote hosts for which it has developed Grid Resource Allocation Management (GRAM) (see § 2.1.1); Data Management, the ability to move data to, from and between servers for which it has developed Global Access to Secondary Storage (GASS) (see § 2.1.2) and modified the File Transfer Protocol (FTP) to produce GridFTP [16]; and Information Management, the ability to describe compute resoures and index this information in a hierarchical way using the Lightweight Directory Access Protocol (LDAP).

Legion

Legion [3] (now Avaki) is an object based middleware. Hosts are abstracted to *host objects*, filesystems to *vault objects* and executables to *implementation objects*. *Binding agents* describe logical ID to physical address mappings and *Context objects* map names given by the user to Legion objects.

UNICORE

UNICORE [17] is a workflow management system allowing jobs split into a workflow of Abstract Job Objects (AJOs) to be executed on a grid. The grid nodes (called U-Sites) consist of three tiers: the Gateway whose job it is to receive the connections and identify the owner of the job, the Network Job Supervisors (NJS) which resolve the AJOs and either forwards them to other NJSs or passes a job to the third component the Target System Interface (TSI), that executes the job.

UNICORE also handles file systems in a special way, introducing *U-space* – essentially the input/output sandbox, *N-space* – the file systems available to the submission client and *X-space* – the (Uni**X**) file system available during execution of the AJOs. UNICORE enables files to be copied between these spaces as and when appropriate.

Others

A number of other technologies with varying degrees of grid-likeness exist. These include Sun GridEngine (SGE) [18, 19] and Condor [20] which are like batch systems or workload managers but with a number of grid extensions. Another example is Netsolve. Netsolve is an agent that advertises a number of libraries available on a system that can be accessed using the Netsolve client. There is a complex data grid called the Storage Resource Broker [21] which has a basic interface. For the Windows operating system a middleware called Entropia^{TM} [22] is available.

The three major grid middleware products highlighted above do not address all issues in the realm of eScience/eResearch. Globus provides a toolkit for building grids whereas UNICORE concentrates on building grids where work-flow is a key factor, and Legion provides an abstract view of a grid and its components. The simplicity of Globus makes it a suitable choice for constructing grids with low level operation in mind (e.g. simple file transfer and job execution). As such, it is a useful starting point when introducing such a rapid paradigm shift in computing to a large user community as is required by the particle physics community (see §1.3).

1.1.3 Web Services

2001 saw the grid community's work starting to overlap with *web service* [23] mechanisms in development by the web community. A number of preliminary discussions (BOFs) were held at the third Global Grid Forum in October 2001 to explore the possibilities of how to utilise these web service technologies.

There was at this time the need to solve certain security problems, for example issues arising from the requirement of grid technologies to open many ports and the conflict with firewall policies of the service providers. Web services also address the level of service provided. A Globus (version 1 or 2) based grid provides only a

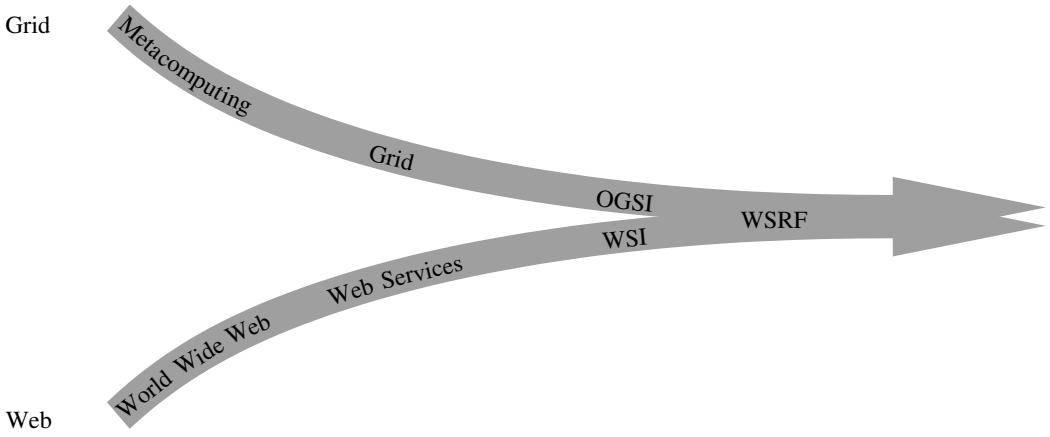


Figure 1.1: The merging of grid computing and web services [9, 26]

coarse set of tools for running code on a grid (i.e. users get full access or no access); web services on the other hand, expose strict remote procedures, access to these is achieved by passing SOAP messages. Web services can be configured to provide as much or as little access to a resource as necessary. Further information on the topic of web services, SOAP, etc. can be found in for example [24].

Precise details on how to align grids with web services has been a topic of hot discussion. At the current time the Web Service Resource Framework (WSRF) is emerging as the preferred way to extend web services to support common patterns that arise in grid computing. However, the details of these specifications are still the subject of debate in the OASIS [25] standards organisation. Therefore the technologies are not quite ready to be used in the production environment.

Figure 1.1 shows how the collision course between the web service community and the grid community seems to have been resolved (at least for the moment).

1.2 Distributed File Systems

We can split data into two different categories. There is a type of data like the event data collected at high energy physics experiments that is read only and contains terabytes of information. In most cases this data cannot be shifted on demand. In this scenario, at the current time, it makes more sense to shift the compute tasks to where the data is.

On the other hand, the compute tasks themselves generally require input files, steering files, metadata and output storage. To get this control data to the compute tasks requires special attention. One approach is to bundle the whole environment into a single archive (e.g. using the Unix `tar` command). Another is to install as much of it as possible at each site. A third is to install the environment in a distributed filesystem.

The evolution of the Grid [27] has this to say about the requirement for distrubuted file systems:

> *Distributed file systems and caching:* Distributed applications, more often than not, require access to files distributed among many servers. A distributed file system is therefore a key component in a distributed system. From an application's point of view it is important that a distributed file system can provide a uniform global namespace, support a range of I/O protocols, require little or no program modification, and provide means that enable performance optimisation to be implemented (such as the usage of caches).

The AFS fits this description remarkably well. It has a global namespace `/afs` and different administrative domains appear as subdirectories of this namespace. It supports any I/O protocol a normal filesystem could, provided the necessary authority has been obtained (locally using normal Kerberos methods, remotely using PAMs[2] or ticket passing). It requires no application program modification save only

[2]Pluggable Authentication Modules - used, in this case, to pass authentication tokens between processes.

a different starting directory. It makes use of a very sophisticated caching system that has been in production over the last 20 years.

From §1.1.2 we see that of the main middleware options available, only the Globus Toolkit®approach does not provide a seamless way to access files: UNICORE has its U-Space, Legion has its vaults.

1.3 Particle Physics

Particle Physics experiments have now become so large and so complex that they require huge amounts of computing resource, scientific analysis, collaboration and funding. The Large Hadron Collider will, when switched on, collect so much data that local online analysis and conventional data storage will be pushed beyond the bounds it currently supports.

1.3.1 The Physics

Why do we need experiments that have such a compute and data intensive requirement? Aside from the socio-economic and political arguments which can be debated elsewhere, there are a number of questions burning in the minds of physicists and astronomers today [28]. Why is our universe inherently made of matter? Why does the model of the universe predict more matter than we can measure? Why do particles have mass? Why is there such a large gap between the strength of the force of gravity and other forces? How indeed do we fit gravity into the Standard Model? Does the SM describe the particles and fields we can detect and measure?

Discovery of, for example, the Higgs particle, SUSY or light extra dimenstions are in the sights of the Large Hadron Collider (LHC) experiments. A deeper understanding of quark–gluon plasmas and of CP violation are also in the design. This

requires new bigger experiments with machine energies and luminosities above those achieved to date. The demands on any such detector in an environment such as this are huge.

1.3.2 Detectors Demands

The Large Hadron Collider at CERN will have four experimental detectors due to be switched on in 2007. The rate at which interactions take place, determined by the rate of particle bunch crossings, will be 40 MHz. At one of the four experiments, this corresponds to an estimated data rate of approximately 1 PB/s. It is estimated that after initial online processing, the rate at which physics events are recorded will be (100) Hz with a typical event size of 1-4 MB. This corresponds to a data rate 100-400 MB/s. That is about 1 CD every 2-6 seconds and networks today are only just able to carry this much data[3]. More details regarding the computational requirements of the LHC can be found in [29].

There are also a number of particle physics establishments currently hosting active experiments that are working on grid technologies. These include DØ at FNAL which has developed SAM[4]-Grid [30] and uses grid technologies for data reprocessing, and the BaBar experiment at SLAC which is developing BaBarGrid. BaBar already distributes its data to three 'Tier A' sites and from there to many 'Tier B' sites. For this it has already developed MetaData Cataloging techniques. Each of these 'Tier B' sites have compute resources associated.

[3]A single faultless gigabit ethernet 802.3 link can move data at a rate of 125 MB/s.

[4]SAM-Grid has two components: Sequential Access through Meta-data (SAM) and Jobs and Information Management (JIM).

1.3.3 The LCG

The LCG is the Lhc Computing Grid [31]. It is a project tasked with taking the European DataGrid (EDG) middleware and testbeds and developing it into a production grid computing environment for the LHC.

The LCG must develop a computing infrastructure that can handle large quantities of data ($\sim$ 1 PB/year) and provide facilities to run large scale analyses over these data. It needs to be accessible to all end-users and so must have straightforward client-side tooling and server-side information. Users must be able to find data, find resources, stage jobs and store results. To this end the LCG will provide User Interfaces, Compute Elements and Resource Brokers, Information Indexes, Logging and Bookkeeping, Storage and Replica Catalogues; all in an online and secure environment.

To handle the huge amount of data reconstruction, data reprocessing, data analysis and simulation the LCG is looking to incorporate $\sim 100,000$ [32] fast PCs which corresponds to the estimates of the processing power required for the LHC alone ($\sim$ 100M SPECInt2000 [33]).

1.4 Thesis Summary

The methods and questions in this thesis were initially aimed at infrastructure for BaBarGrid. However, the technologies used in this thesis are robust and have undergone a great deal of testing in the wider community. The middleware developed in this thesis is designed to meet ease-of-use requirements driven by users of the system.

The rest of this thesis takes the following form. Chapter 2 describes technologies used to provide AFS access to jobs submitted in a Globus Toolkit®grid. It covers

the transfer of authentication tokens to allow this to happen securely. Chapter 3 investigates whether there is any serious impact on performance when using AFS compared to other methods used in distributed environments. Chapter 4 described the middleware developed to bring the necessary technologies together. Chapter 5 describes how extra monitoring for jobs was developed. Chapter 6 covers the application of these technologies to BaBar analyses. Following this there are discussions and conclusions. As this is essentially a multi-disciplinary thesis a glossary has been provided immediately after the references list.

Chapter 2

Technology Review

This thesis explores the use of AFS as a solution to problems associated with file IO in a Globus Toolkit®based grid. The following chapter discusses in detail how relevant parts of these two technologies work and how they can interoperate using a third piece of software.

Two further technologies are introduced that were incorporated into the middleware developed for this thesis; one to allow continued access to AFS, the other to monitor the job's status.

2.1 Globus Toolkit®

The High Energy Physics community have adopted the Globus Toolkit®version 2 based middleware for job submission. This is conceptually different to the current leading edge technologies that adopt a Service Oriented Architecture (SOA), Globus Toolkit®version 3 being an example. Such web-service based technologies are outside the scope of this thesis and are not reviewed below.

Of all the components supplied by the Globus Toolkit®, for this thesis it is only

necessary to review two, GRAM (job submission) and GASS (file transfer). The need for the latter is not immediately obvious, given the use of a global file system. As will be made clearer in subsequent chapters, we need the ability to transfer and run a small script to set up access to the global file system, which is not granted automatically by the job submission service[1].

The next two subsections review these two Globus services.

2.1.1 The GRAM

The Grid Resource Allocation Manager [34] (GRAM) allows a job to be submitted from a client or agent to be received and scheduled/executed on a server. It uses an HTTP based protocol which passes messages pertaining to job descriptions. A job is described using Resource Specification Language [35] (RSL).

The GRAM parses incoming requests received, authenticated and authorised by the server's gatekeeper. The gatekeeper forks a child process, and changes the ownership of that child to that specified by the *gridmap-file*[2], which then executes a *jobmanager*.

The GRAM handles the job from submission through staging-in, queuing, running, checkpoint/restarting, staging-out, to the end of the job.

Jobmanagers

Jobmanagers are essentially scripts that interface an incoming Globus request to a batch system. The jobmanager parses the incoming RSL and creates the corresponding batch queue submission script. There are jobmanagers for LSF [36], PBS [37],

[1]The alternative, of modifying the Globus server to set up access to the global file system, puts extra pressure on the system administrator at each site.

[2]The gridmap-file is a reference file that helps Globus services identify local user accounts from the security context in the Globus communications.

NQE [38], GridEngine, Condor and *fork*. The fork jobmanager, if present, will allow the submitted job to run immediately on the gatekeeper node as a child process of the jobmanager.

Each jobmanager continues to run on the gatekeeper node while the user's job runs locally or elsewhere. This allows it to receive queries and commands from the user such as *poll*, *cancel* and *refresh credentials*. The last is useful when a submitted job has been queued for some time and valid credentials are required for the duration of the job (e.g. when using AFS).

RSL

Resource Specification Language (RSL) is the specification for Globus jobs that, in most cases, allows the job submitter to indicate to the Globus jobmanager, certain job and queue attributes. It provides a common interface to the batch systems targeted by the jobmanager. Key-value pairs are conveyed to specify estimated time requirements, process multiplicity, input/output redirection, environment variables, working directories, etc. Values may contain single or parenthetically, multiple entries.

More advanced RSL features can be exploited for multiple resource jobs and complex requirements. Further information is available in [35, 39].

2.1.2 GASS

The Global Access to Secondary Storage service [40] is designed to provide file access across a grid environment. It provides server software that is available to both the client and the grid servers. It utilises security mechanisms provided by the basic Globus installations. Data connections are made between ephemeral ports via control commands usually through the Globus gatekeeper. During a Globus

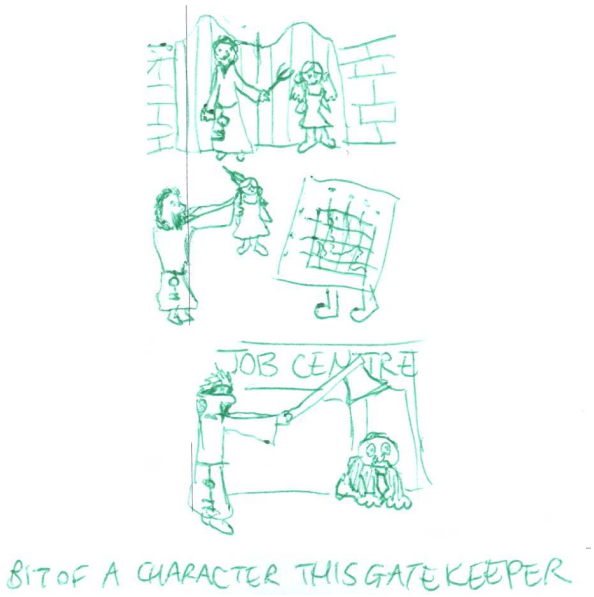


Figure 2.1: Artists impression of GRAM

job submission a GASS server is set up on the client machine. This allows the execution program to be staged and if the interactive mode is used (e.g. by using `globus-job-run`) the standard output and error streams are redirected through GASS back to the client.

There are of course firewall and Network Address Translation (NAT) issues associated with this type of communication. Globus clients must choose to either open up their firewall or open part of it and specify an environment variable for Globus to use in port negotiations.

2.1.3 The GSI

The 'Grid Security Infrastructure' (GSI) was developed for the Globus Toolkit® and is based on the public key infrastructure. They provide extensions to the Public Key Infrastructure (PKI) that allow for secure single sign-on and reasonably secure delegation. The GSI is discussed in detail in the security section 2.3.

2.2 The AFS

The AFS, formerly the Andrew File System, presents itself to the client's operating system as a mounted file system. The AFS client works with a small hook in kernel space (the kernel module `libafs` or `nnpfs` depending on implementation) that intercepts filesystem calls directed at the mounted AFS filesystem and passes them to the *cache manager* in user space. It is the job of the cache manager to determine whether to move files across the network, use the cached copies or whether any cached files need to refreshed.

The cache manager is a sophisticated program that has a number of responsibilities. Firstly, it keeps copies of recent files from the AFS servers to reduce network traffic. It also caches directory level information and keeps a record of its status. It registers files and directories in its cache with the AFS server such that a change to files on the server are reported back to the cache manager via a callback mechanism. It also has a limited resilience to certain failures[3]; after a server outage it will re-establish connection and check the local cache against what is registered with the fileserver. Further resilience is achieved by making all communications, to the AFS server, Remote Proceedure Calls (RPCs) making the protocol more asynchronous.

2.2.1 The File System Structure of the AFS

The AFS is designed to look like a UNIX file system, such that normal file system operations — create, delete, rename, open, close, read, write, append and seek — behave transparently to running processes. Get and set attributes appear to behave as expected; however, access is controlled by the AFS rather than the standard Unix permissions (see §2.2.2).

The whole AFS filesystem is mounted in a global namespace `/afs`. Directly

[3]Failures have been observed on the windows AFS client where the return status of close(3) is not checked.

under this mountpoint (in a read only directory) are the *cell* mountpoints. These are pointers to root volumes on all configured AFS cells. The cells are configured by the client and refer to AFS fileservers somewhere on the internet. The cells represent different administrative domains[4] [42].

Access to two different cells on different servers requires two different AFS service tokens (Kerberos AFS tickets see §2.3.1), although these may be obtained from the same Kerberos server.

Beneath the root volume level, eg `/afs/hep.man.ac.uk`, entries correspond to *volumes* or date in the root volume. A volume is like a mounted partition. It corresponds to a data storage unit on the AFS fileserver. Volumes can be moved between replica AFS fileservers without impact to the client.

2.2.2 Access Control

Access to files on an AFS server is controlled by the logical OR of the Unix user permissions and the AFS access control settings. An AFS Access Control List (ACL) contains a list of users, groups, negative and positive rights for a combination of various elements of access. These elements are shown in table 2.2.2.

The AFS ACLs were the inspiration behind the Grid Access Control Lists [43] (GACL) used in GridSite (§2.3.6) for authorisation for file access on a web server.

Access Control to the contents of a cell is controlled through that AFS server's Protection Server (PTS) and recognises individual entities (principals) as well as groups. Access to cached data and AFS tokens is controlled by the cache manager and is based on either the Unix UID or Process Authentication Group (PAG) [5]. The following groups usually exist: "system:anyuser" – anyone in the world with an AFS

[4]Different cells may be in the same Kerberos [41] realm but authority still needs to be obtained for each. Kerberos is discussed further in §2.3.1

[5]The PAG information is a property of the kernel.

r	Read:	Allows bytes in a file to be examined.
l	List:	Allows the directory to be decended into, and allows file metadata to be examined.
i	Insert:	Allows files/directories to be created.
d	Delete:	Allows files/directories to be deleted.
w	Write:	Allows bytes to be placed in a file.
k	Lock:	Allows file access to be locked.
a	Admin:	Allows the ACL to be modified.

Table 2.1: AFS access control list elements

client (no encryption/authentication is needed for access), "system:administrators" and "system:authuser" – like system:anyuser except that the client must be known to the AFS server and transferred data is either authentic or encrypted and authentic.

2.3 Security

This section discusses the security models adopted by the AFS and the Globus Toolkit®. First a review of Kerberos is provided. This is followed by a summary of the Public Key Infrastructure. A discussion of why these technologies are deemed non-complete for the grid environment is presented. Then the GSI is described and the differences between the GSI and the PKI are highlighted.

In order to use AFS in a Globus Toolkit®based grid a crossover between the two securities is required. A description of GSI enabled `klog` is provided. K.X509, which provides a similar crossover of technology in the opposite direction, is not discussed in this thesis but may be found in [44].

Finally, the MyProxy [45] service and GridSite Apache module are discussed.

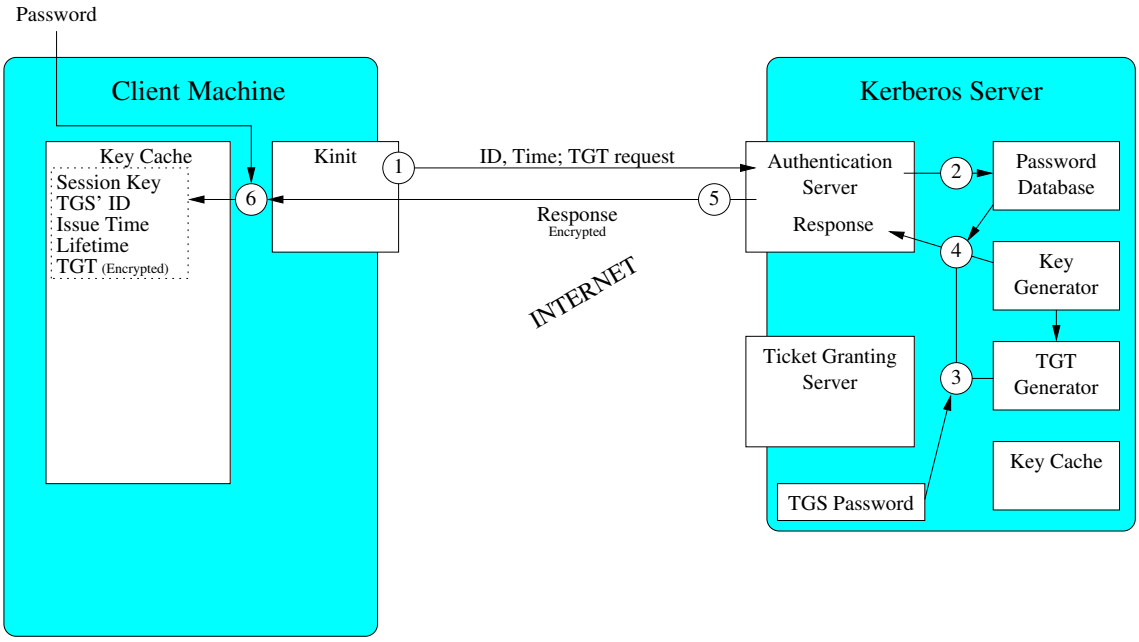


Figure 2.2: Kinit, Authentication in Kerberos IV

2.3.1 Kerberos

Kerberos comes in two release versions, Kerberos IV and Kerberos V. Version V fixes a few security holes present in version IV. This section introduces version IV.

Figure 2.2 shows the first stage of the Kerberos IV security mechanism. In step ①, the user issues a kinit command. This checks the configuration or the command-line for the details of the Kerberos IV server and user's *principal* (their identification in that Kerberos IV *realm*). The message sent to the server contains the principal, and time. The Kerberos IV Authentication Server (AS), having received a request for a Ticket Granting Ticket (TGT), checks the password database, ②, for an entry corresponding to that principal. If it exists a DES[6] key is generated. A ticket granting ticket is generated using the system time, a lifetime, the identity of the Ticket Granting Server, the IP address of the client, the identity of the client and

[6]Data Encryption Standard; DES is a symmetric cipher based on a 56 bit key.

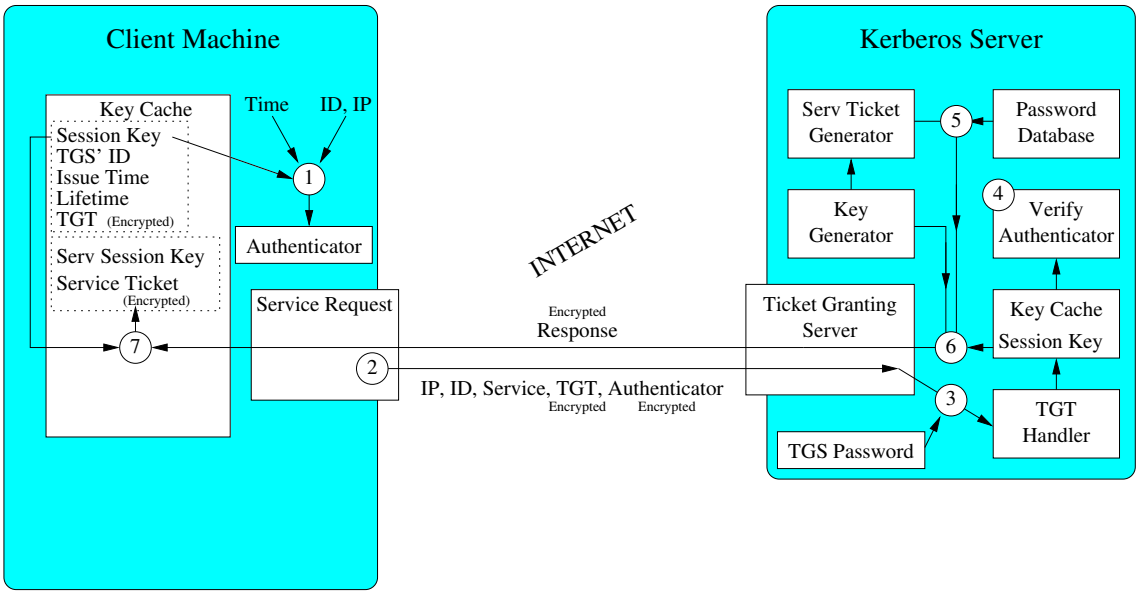


Figure 2.3: The TGS, first stage authorisation in Kerberos IV

the key that was just generated. The TGT is then encrypted, ③, using the TGS's 'password' (being a strong secret shared only between the TGS and the AS). The encrypted TGT and its lifetime, the session key, the identity of the TGS and the time are all encrypted with the client's 'password', step ④. In step ⑤, this bundle is sent back to the client. Only the client can makes sense of it; others can only decipher and use the response if either they know the password ⑥, or can crack the password within the lifetime of the ticket. The lifetime is chosen and is much less than the estimated time required to crack a user password. The decrypted results are stored by the client.

The second stage of the mechanism, as illustrated in figure 2.3 is a request for an authorisation token (Server Ticket). Firstly, an authenticator is generated ①. The authenticator consists of a time stamp, the principal, the IP address, a short ($\sim$ 5 minutes) lifetime and optionally an issue number (to stop replay attacks). This is all encrypted with the session key. The name of the service (that the client wants to use), the TGT and the authenticator are sent (②) to the Ticket Granting

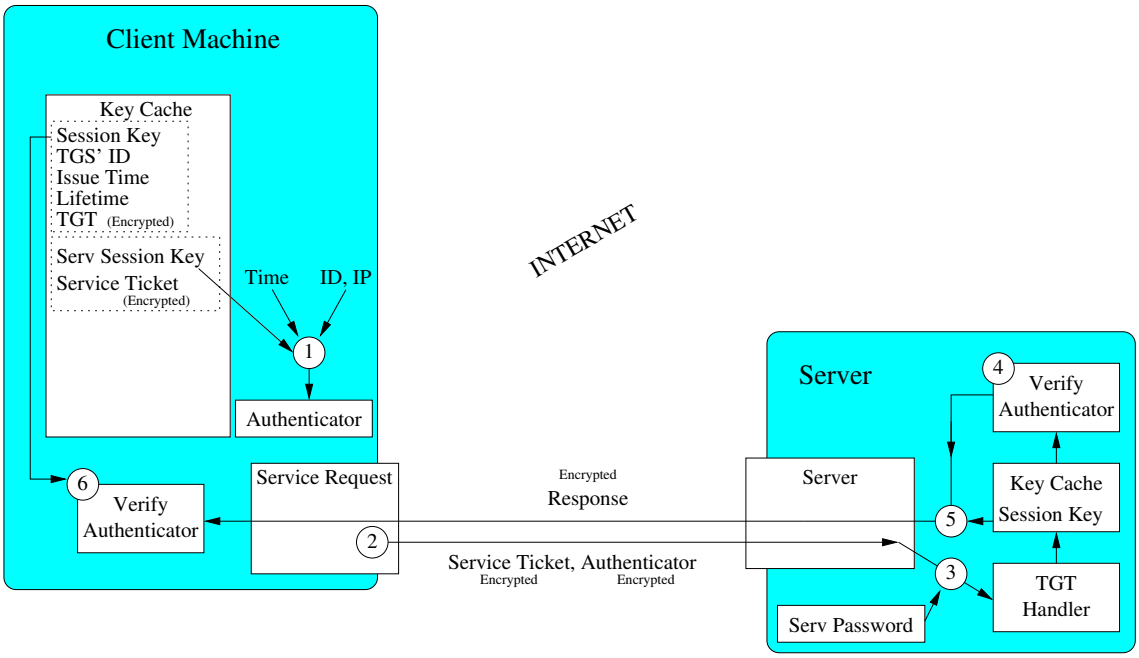


Figure 2.4: The service, authorisation in Kerberos IV, second stage

Server (TGS). The TGS decrypts the TGT with its 'password', ③. It extracts the session key and uses it to decrypt the authenticator. The information carried by the authenticator is checked ④ (successful decryption proves authenticity). A new *server* key is generated. The corresponding server ticket is generated using this key and encrypted ⑤ using the 'password' (shared between the service requested and the TGS). This service ticket and the new session key, the identity of the service, a timestamp, and a lifetime are encrypted ⑥ with the first session key and returned. The response is decrypted by the client using the first session key, to reveal the service ticket and the service key.

To conclude the authorisation step the server ticket must be presented to and verified by the server. Moreover, the response from the server needs to be checked by the client to achieve mutual authentication. Shown in figure 2.4 is a schematic of the communication with the server. The client makes another authenticator ① using the session key obtained from the TGS, and sends it ② along with the service

ticket to the server. The service decrypts the service ticket using its 'password' ③ to obtain the session key with which it decrypts and verifies the authenticator ④. The server then modifies the authenticator in a standard way and encrypts it using the session key ⑤ before passing it back to the client. The client receives the modified authenticator and decrypts it to check the authenticity of the server.

For the AFS the whole process is wrapped up in one or two commands `klog` (figure 2.2 and 2.3) or `kinit` (2.2) and `afslog` (2.3), depending on implementation. The AFS server authorises access based on these credentials and the ACL as necessary (figure 2.4).

2.3.2 The PKI

The Public Key Infrastructure (PKI) deals with the key exchange problem differently. Using public key cryptography as a means of authentication in an online system it is possible to reduce the online dependency on the source of authenticity, the Certificate Authority (CA). The identity of a client (or server) is ascertained by proving that they hold the private key of a pair of keys. They present the public part during the handshake. The public part is contained within an X.509 [46] certificate which is verified by the other party.

Only the private key is able to decrypt messages encrypted with the public key and vice versa. Therefore, if during the handshake phase the server encrypts a message with the client's public key and the client returns that message securely to the server, then the server knows that the client has the private key. The same process can be done with the server's public and private key, providing mutual authentication. Alternatively, one part may present a message, the other *sign* it with their private key and return it. Authenticity is ascertained when the message is decrypted with the public key.

The advantage of this method comes when a third party can encrypt a copy of the

client's and the server's public keys along with some other identifying information with the third party's private key. If they do this in response to having assertained the identity of each party (and the client and server trust this activity) then the client (server) only needs to have the public key of the third party to assertain the identity of the server (client). This is the rôle of the CA.

An X.509 certificate is created by the CA which constructs it from a *certificate request* and signs it either with private key of a self-signed X.509 certificate or the private key of another valid X.509 certificate. Certain restrictions apply in versions 2 and 3 of the X.509 standard.

Figure 2.5 shows the contents of an X.509v3 certificate in human readable form. The certificate contains amongst other things, the issuer's identity, the validity period, the *subject*, the public key and CA's signature. The subject is used to tie the private/public key ownership to an individual or server. The issuer is used to locate the signing certificate from any list of certificates that are supplied.

Trust is achieved when a certificate used to establish a connection is signed by a trusted CA. It can also be reached by having any number of intermediary CA X.509 certificates in a chain (one signed by the next) as long as one CA in the chain is trusted.

Figure 2.6 shows a basic full mutual authentication using the PKI. In this case the purpose of the handshake is to provide each party with the other's public key and set up a secure connection at the socket level.

The client, ①, *signs*[7] a message containing the time, a one time only message (*nonce*), the server name and their public key. The server receives the message and after checking that it is the addressee, verifies the client's public key against the list of CAs that it trusts, ②. The server creates a symmetric key and encrypts it, ③,

[7]Signing involves taking a hash (a one way reduction of the message using for example the MD5 or SHA1 algorithm.) and encrypting it with the private key.

```
Certificate:
   Data:
       Version: 3 (0x2)
       Serial Number: 2503 (0x9c7)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=UK, O=eScience, OU=Authority, CN=CA/Email=ca-operator@grid-support.ac.uk
       Validity
           Not Before: Aug 11 14:11:50 2004 GMT
           Not After : Aug 11 14:11:50 2005 GMT
       Subject: C=UK, O=eScience, OU=Manchester, L=MC, CN=michael jones
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:dd:7b:68:79:52:ec:5c:a2:31:77:5b:9d:9e:78:
                   c9:d7:1d:e2:79:f5:4e:2e:02:dd:b5:f8:42:ef:69:
                   72:f4:44:53:3a:d7:48:08:e3:84:eb:d0:e9:9c:cc:
                   57:20:2d:3c:65:11:61:c6:20:ea:89:1a:3d:30:a8:
                   61:4a:3e:68:eb:b3:70:0a:d1:0e:08:95:2d:ef:d4:
                   85:56:a7:40:21:f7:25:51:3c:97:2b:d6:97:73:2b:
                   a1:0a:5a:ce:3a:a7:ef:c8:09:61:a7:28:55:da:73:
                   db:18:a4:36:a2:76:83:d9:eb:ee:9d:7b:f1:62:28:
                   da:c5:6e:4f:cb:17:44:3d:41
               Exponent: 65537 (0x10001)
       X509v3 extensions:
           X509v3 Basic Constraints: critical
               CA:FALSE
           Netscape Cert Type:
               SSL Client, S/MIME
           X509v3 Key Usage: critical
               Digital Signature, Non Repudiation, Key Encipherment, Key Agreement
           Netscape Comment:
               UK e-Science User Certificate
           X509v3 Subject Key Identifier:
               FB:1F:8C:93:39:73:FE:AC:31:DC:CB:72:0F:A9:35:2C:7A:12:A0:93
           X509v3 Authority Key Identifier:
               keyid:02:38:AB:11:A3:96:80:8B:0D:D3:15:2B:08:A5:8E:30:DA:B2:DA:A8
               DirName:/C=UK/O=eScience/OU=Authority/CN=CA/Email=ca-operator@grid-support.ac.uk
               serial:00

           X509v3 Issuer Alternative Name:
               email:ca-operator@grid-support.ac.uk
           X509v3 Certificate Policies:
               Policy: 1.3.6.1.4.1.11439.1.1.1.1.4

           Netscape CA Revocation Url:
               http://ca.grid-support.ac.uk/cgi-bin/importCRL
           Netscape Revocation Url:
               http://ca.grid-support.ac.uk/cgi-bin/importCRL
           Netscape Renewal Url:
               http://ca-renew.grid-support.ac.uk/renew.html
           X509v3 CRL Distribution Points:
               URI:http://ca.grid-support.ac.uk/cgi-bin/importCRL

   Signature Algorithm: md5WithRSAEncryption
       37:13:f3:93:85:eb:05:e8:80:00:d5:78:c8:9e:d0:16:f5:b7:
       42:42:4b:16:10:7c:dd:81:fa:ec:b5:f8:d5:26:db:10:60:b1:
       55:5c:a6:f5:56:8c:8a:48:05:0a:1d:da:b0:f7:b8:15:6b:71:
       70:65:70:94:63:d3:e5:a8:80:e0:b0:e9:11:2e:ec:c8:bc:31:
       9e:50:f4:23:26:3d:b7:e2:c4:cd:1a:cf:db:76:af:05:4c:9b:
       37:e0:83:0c:62:66:2f:8c:39:29:01:74:0d:a8:be:3c:19:df:
       61:c0:7f:9c:20:4c:5b:f8:f0:7b:7b:d1:c5:55:b0:b0:af:b9:
       1f:d9:c7:47:dd:33:ac:90:87:31:2a:30:35:6e:31:17:16:76:
       96:be:05:06:aa:0a:35:81:2f:b0:c4:15:3d:0a:dc:76:b3:fe:
       d4:d9:c6:6a:92:12:36:ba:48:94:62:db:71:b0:3c:ec:8c:ed:
       99:2d:42:43:ef:d7:7e:de:03:db:97:e2:6d:a9:cb:e0:ec:3f:
       88:d1:ca:a4:b7:1d:79:db:fb:57:15:98:b2:cf:c8:e8:d3:cf:
       be:1c:23:67:40:46:89:42:c2:f0:47:51:c5:66:1a:b3:fb:1a:
       e4:79:52:74:0f:e5:56:3c:25:78:a5:54:3f:f9:4b:d6:f5:3e:
       eb:9c:43:8a
```

Figure 2.5: A parsed UK eScience X.509 certificate

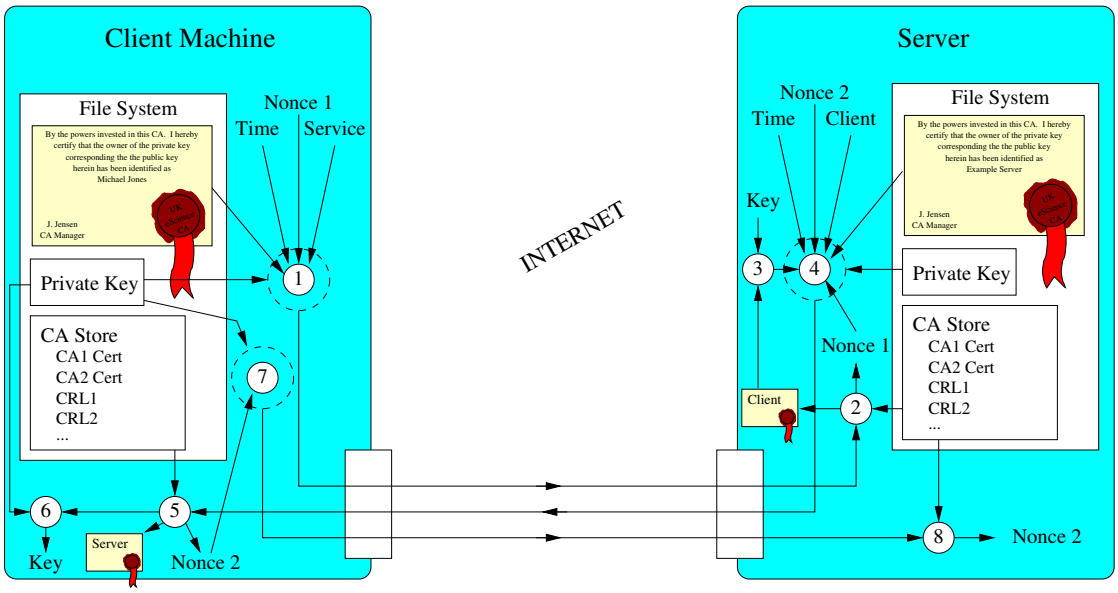


Figure 2.6: PKI authentication

with the client's public key, and places it in a message, with the original nonce, a new nonce, the time and the client. It signs the message, ④, and passes it back to the client. The client unpacks the message and verifies it in the same way, ⑤, extracting the key, ⑥, signing and returning the second nonce, ⑦. When the nonce is received by the server ⑧, the signature is verified. Both parties now know that the other posesses the private key corresponding to the public key they sent.

The exchanged public keys are verified by the CA's signature. If the CA that signed one party's certificate is trusted by the opposite party and vice versa, then their identities are now known to each other. The symmetric key passed in the response (encrypted using the initiator's public key) can now be used to set up the secure channel.

CRLs

A Certificate Revocation List (CRL) is an additional piece of security information that can be used in PKI. If a private key is compromised the owner can tell the issuing CA. That CA can then publish this certificate's serial number in a CRL. Servers and clients can download this list periodically and verify its signature. Once its validity is positively verified, the server (or client) can make an extra check during all certificate verification processes. The CRL carries a lifetime constraint so that if it expires the server or client may choose not to trust any certificate issued by that CA.

2.3.3 The GSI

Neither the PKI nor Kerberos were deemed sufficient for the Globus Toolkit® [11]. Kerberos was rejected because it required a global naming convention and too much server interaction. PKI was rejected because it did not provide for delegation and single sign-on. As a result, the Grid Security Infrastructure [47] (GSI) was developed.

The GSI is virtually identical to the PKI. In the most basic form the X.509-based *Impersonation Proxy Certificates* [48] used are similar in all respects to the end entity certificates used in PKIs except that it is an end entity's certificate (or *leaf certificate*) that is used to sign another certificate. This is not allowed in X.509 versions 2 and 3, and means that most SSL enabled servers (e.g. webservers using HTTPS) will fail to authenticate an incoming connection based on such a certificate. It is possible to modify these servers such that they will accept proxy certificates and a discussion of one such modification is described in section 2.3.6.

X.509 certificates are generally valid for a long time ($\sim$ 1 year). In comparison, GSI proxy certificates are not. The purpose of a GSI proxy certificate is to allow a process to use the certificate rather than the user. This means that the process must

be able to read the private key of the GSI proxy credential. Keys corresponding to X.509 certificates are usually encrypted and only unlocked with a passphrase. Keys for GSI proxy certificates are usually stored unencrypted with only file permissions protecting them. This is the reason for the short lifetime.

2.3.4 GSI Klog

GSI klog consists of a client program and server dæmon implemented as `gsiklog`[8] [49]. The client side uses a GSI proxy credential to communicate over SSL [51] with a server. The server has access to the Ticket Granting Server's secret key on an AFS server and can therefore issue AFS tickets as if it were the TGS itself. Authentication and authorisation for the issue of the AFS ticket is established using the GSI and checking the entries in the gridmap-file.

Figure 2.7 shows a schematic of the `gsiklog` process. Using the GSI a secure socket is created. A ticket request for the AFS service similar to the Kerberos request to the Ticket Granting Server (figure 2.3) is sent. The request contains the Kerberos realm, the AFS server's ID, the request lifetime and (optionally) the principal and the instance.

The server performs a look-up of the distinguished name of the client from the GSI Credential and if requested the principal. If there's a match, a ticket is generated with the lifetime. The lifetime is limited by the server configuration, the GSI credential and the request. The session key, principal name, server name, lifetime, ticket and ticket version are returned via the secure socket.

[8] or the later version, `gssklog` [50], which uses a slightly different protocol

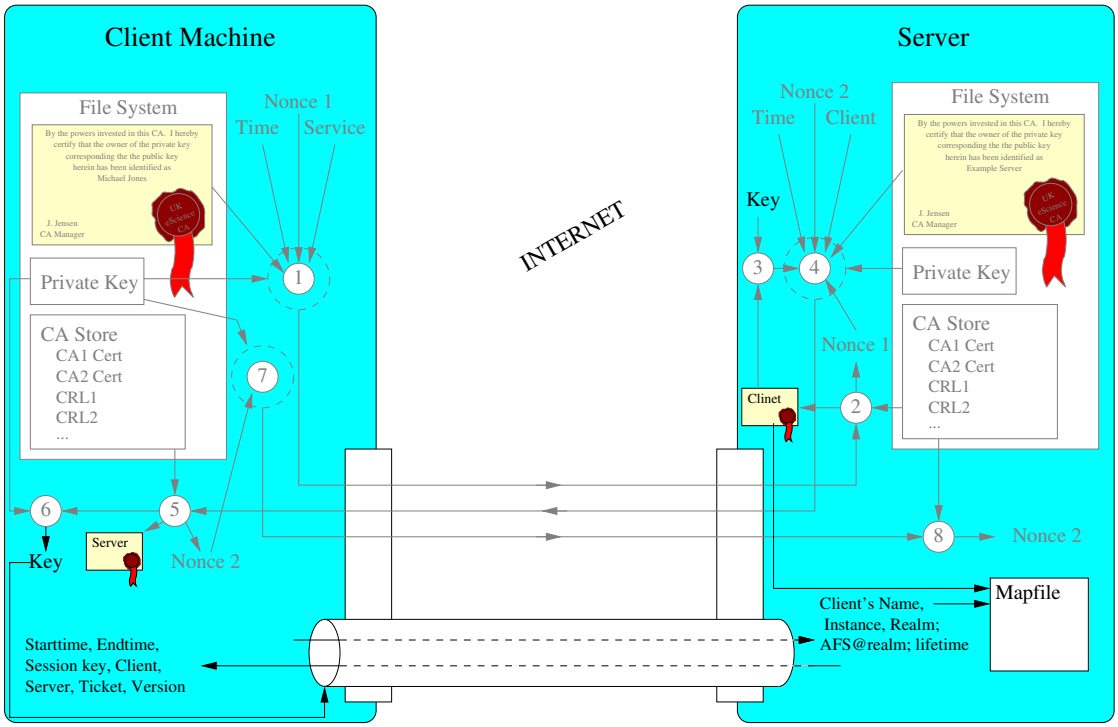


Figure 2.7: GSI enabled `klog`

2.3.5 MyProxy

MyProxy [45] is a medium-term credential storage server. It is designed for the purpose of storing GSI impersonation credentials (with a lifetime $\sim$ 1 week) and delegating GSI credentials to authorised processes on the internet. The system splits the GSI signing process into two, to fit into a client-server model.

The motivation for MyProxy comes from a portal perspective. Using MyProxy allows portals to obtain GSI proxy credentials without sending them across the wire[9]. The first BaBarGrid demonstrator [52] used a GridSite version 0.2 (see §2.3.6) enabled webserver to initiate a Globus-controlled job submission to a grid. The GSI proxy in this case was uploaded using a simple CGI script across a secure https connection.

The client uses a standard GSI proxy (full, limited or restricted[10]) and issues a `myproxy-init` command, which registers a username/password or GSI credential name with the server, and causes it to issue a certificate request for a GSI credential from the client. The `myproxy-init` process signs the request with the local proxy certificate and returns the signed request to the server. The server combines the private key and signed request to create a delegated proxy certificate on the MyProxy server.

A process on a node somewhere on the internet that knows the server, username and password (or has the necessary GSI credential), can request the delegation of a new credential. It does this by using the `myproxy-get-delegation` command. The delegated credential's properties (i.e. lifetime, restrictions and limitations) are controlled by the `myproxy-init` process.

The delegated GSI proxy on the MyProxy server can be deleted by issuing a

[9]Sending private keys across the network is not favoured by the community despite strong encryption SSL and short lifetime proxies.

[10]Restricted proxies are currently only meaningful in GT3+ or RFC3820 compliant services [48].

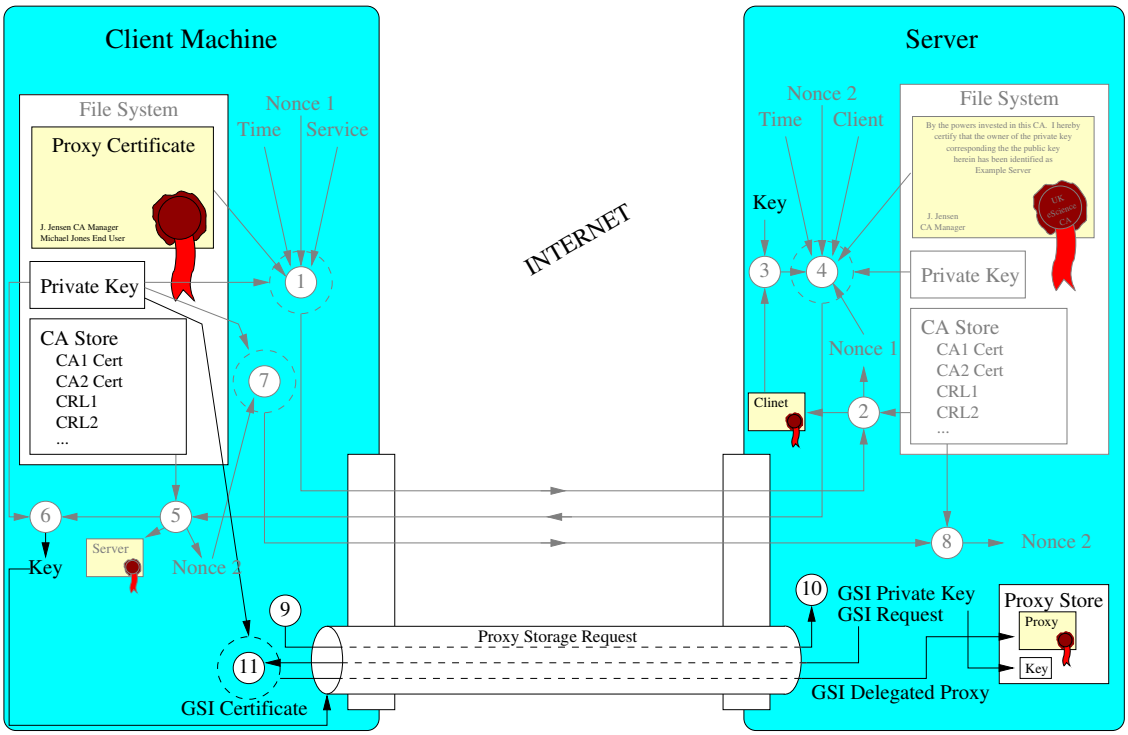


Figure 2.8: MyProxy initiation

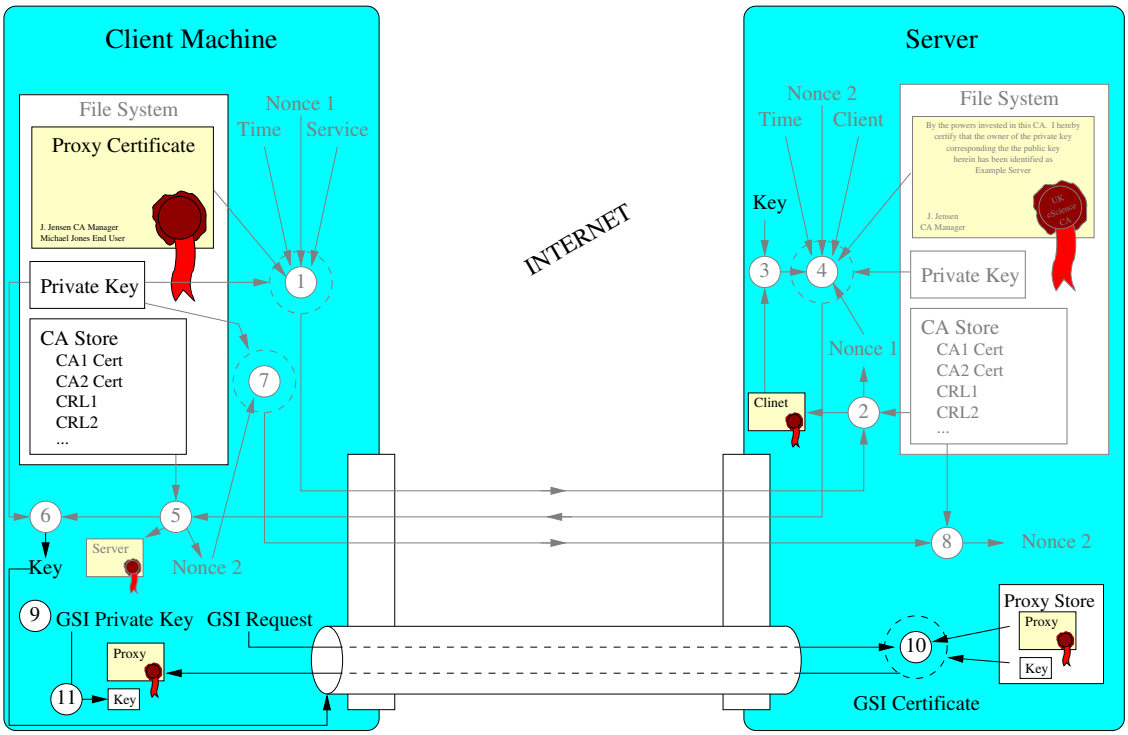


Figure 2.9: MyProxy get delegation

`myproxy-destroy` command. This stops any further delegation from the server, of credentials that correspond to a particular username.

MyProxy is used by the `gsub` middleware described in chapter 4 in order to manage the problem of AFS token and GSI credential expiry in jobs submitted to a grid.

2.3.6 GridSite

GridSite version 1 [53] is an Apache version 2 module. Previous versions were a mixture of `CGI` programs and `mod_ssl` [54].

GridSite was designed to allow the use of X.509 certificates as authority to enable users to modify web sites from their favourite browser. Authority to edit pages could be controlled by the management of a virtual organisation rather than fine grained control at the server.

A browser with access to a public and private key can authenticate to the web server with the `mod_ssl` module enabled, using HTTPS (secure HTTP [55, 56] over SSL [51]). The author of this thesis modified `mod_ssl` to GSI enable it for the SAMD project [57]. Although the code to check GSI credentials is relatively short (360 lines), the patch to produce `GSI-mod_ssl` (Available at the SAMD website [58]) from vanilla `mod_SSL` needs to incorporate the whole certificate checking algorithm to call out to the GSI checking component at the right time.

The GSI extensions allow processes to authenticate to the webserver on behalf of the user rather than requiring the user to unlock their X.509 certificate for each communication. GridSite 1.0 incorporated this patch into the module, during the development of the `htcp` (HyperText Copy) software.

GridSite is used by the software developed for this thesis for the purposes of job and grid-fabric monitoring. Jobs submitted to a grid using `gsub` record their progress

and status to a GridSite enabled webserver using a command line HTTP/HTTPS executable: `curl` [59]. Command line tools to control job submission and monitoring were also written (see chapter 5).

Chapter 3

Performance Study Of AFS

This chapter introduces the AFS protocol. A number of standard benchmarks are described, performed over the wide area network and presented. Data taken from a number of straightforward file transfers is shown, and comparison of the performance of AFS against vanilla GridFTP is also performed and shown.

Application specific tests are performed in chapter 6.

3.1 AFS

The AFS is built upon Rx [60] which is an application level Remote Procedure Call protocol layered on top of the User Datagram Protocol (UDP) [61]. UDP is a packet-based connectionless protocol. The Transmission Control Protocol (TCP) [62] (which is used for most other file transfers) on the other hand, is a stream-oriented connection-based protocol. The data transferred using TCP is broken up into TCP packets, transmitted, monitored and reconstructed by the TCP stacks at each end of the connection. UDP is not concerned with the packets transferred. Both the TCP and UDP headers contain a checksum to check data integrity. In UDP, a failed

checksum means that the packet is dropped whereas in TCP the receiving agent will request that the sending agent resends the corrupt packet. UDP transmits data as it becomes available regardless of network congestion.

The Rx protocol therefore must bridge the gap between unreliable[1] and reliable data transfer. However, because of this it has more control over the data transmission and its implementation is therefore tuned to the requirements of AFS (see e.g. [63]). It does not need to hold connections open or pass through an expensive connection renegotiation as TCP does. UDP itself is not subject to the congestion control that TCP suffers from in long sustained transfers (see e.g. [64]). It also has the facility to place security into the packet headers themselves.

The following subsection starts with a measurement of the performance of AFS. AFS is currently in place and used as the user-space file system on a number of farms in the High Energy Physics Community. The author has access rights to five of these; where possible tests have been made to all five.

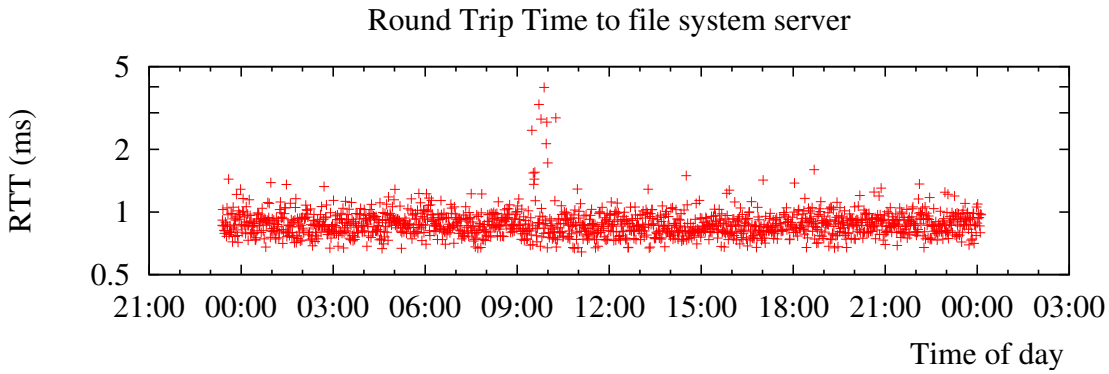
3.1.1 The Andrew Benchmark

The Andrew Benchmark [65] was designed to quantify the performance compromises of switching to the AFS from e.g. NFS. It consists of five phases: MakeDir, Copy, Scan, ReadAll and Make.

- Phase 1 (MakeDir) creates a directory tree from a source tree.
- Phase 2 (Copy) recursively copies the directory tree.
- Phase 3 (Scan) makes status calls on each item in the tree.
- Phase 4 (ReadAll) examines each byte of each file in the directory tree.
- Phase 5 (Make) runs `make` on the directory tree compiling a specific program.

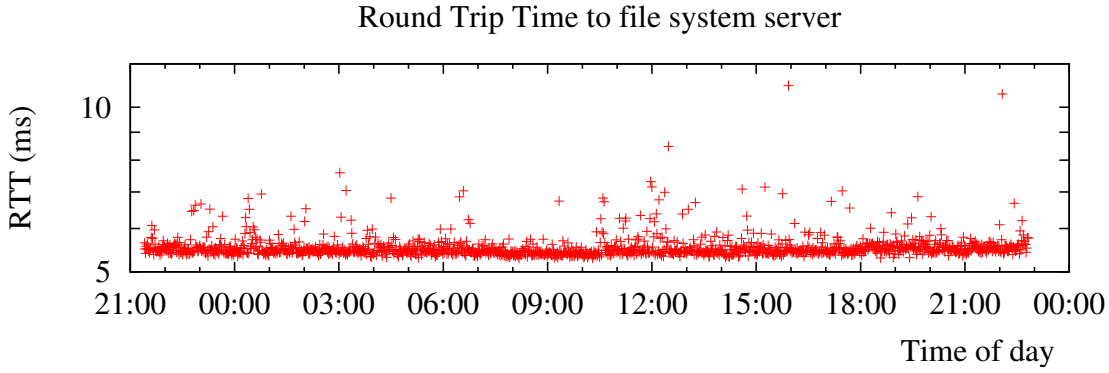
[1] This is not to say that UDP transfers are lossy; this is generally not the case.

A copy of the original Andrew Benchmark is available in [66]. However, Phase 5 no longer builds on modern Linux distributions. A modification to the benchmark is also available from [66]. The modified test uses an `openssh` source bundle instead of an old windows manager code and a small time program. A newer version of openssh was used and a patch was made to the timing program for use in this thesis.



$\overline{RTT} = 0.89 \pm 0.19$ ms

Figure 3.1: Round trip time to the local AFS server



$\overline{RTT} = 5.55 \pm 0.33$ ms

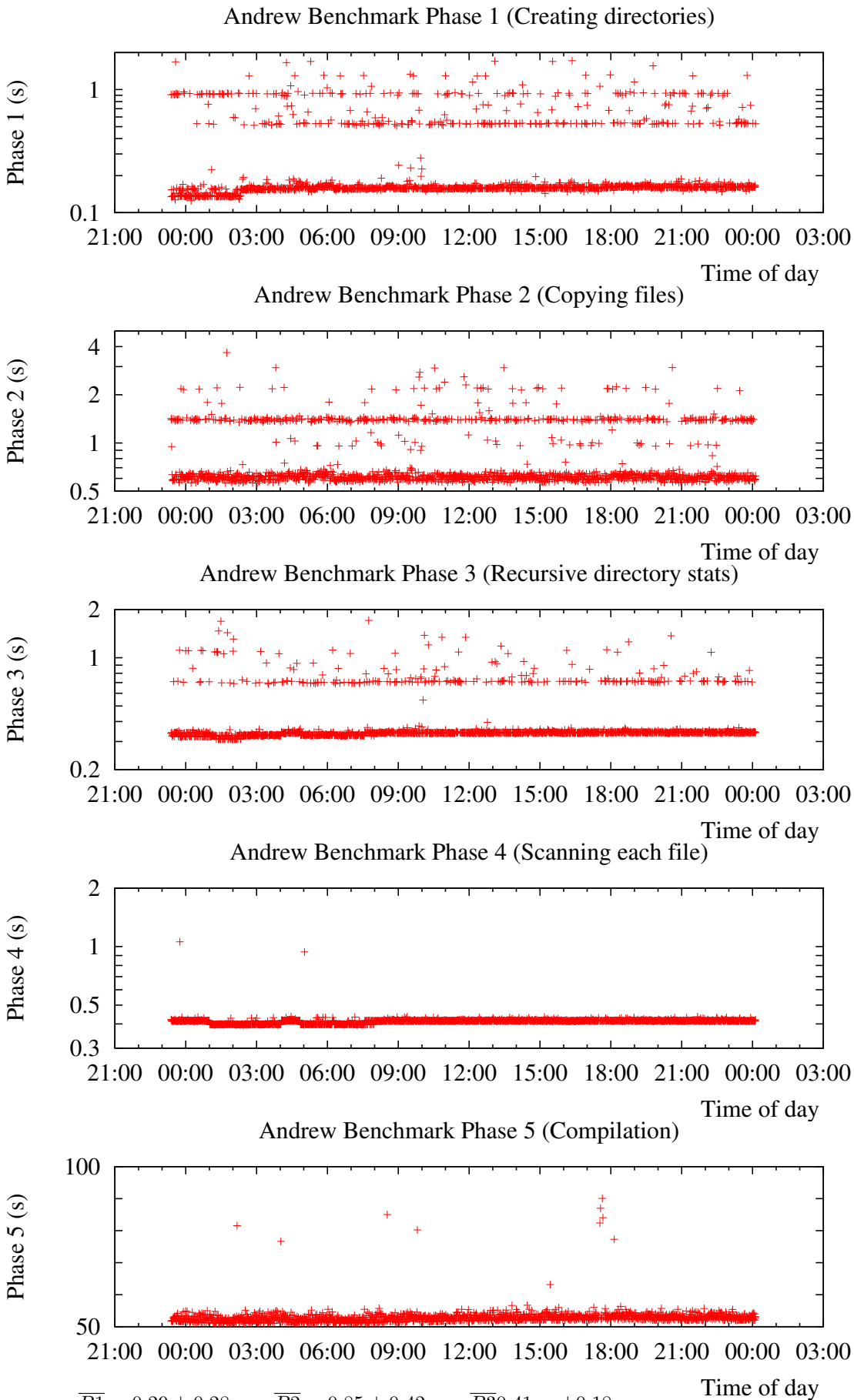
Figure 3.2: Round trip time to a remote AFS server

Six measurements were taken many times over a 24 hour period. The first measurements, figures 3.1 & 3.2, show the round trip time (RTT) as measured using the unix command `ping`, to a local and a remote AFS server respectively. The RTT is a basic measure of latency between the client and server. It is a measure of the network speed over the IP layer. The RTT is included here for illustrative purposes, to provide an idea of the network distance between the AFS client and server.

The observed spread in figures 3.1 & 3.2 may be caused by any number of factors. The most probable cause is the NIC waiting for other transmissions on the local area network to finish (see for example [67]). The spike in figure 3.1 at 10 am may correspond with increased activity at the start of the working day (data shown in figure 3.1 were taken on a Friday whereas data shown in figure 3.2 were taken on a Sunday). The spike at 4 am in figure 3.2 corresponds to the client's (and possibly the server's) default Redhat Linux installation's daily (4:02 am) `crontab` executing system database updates etc.

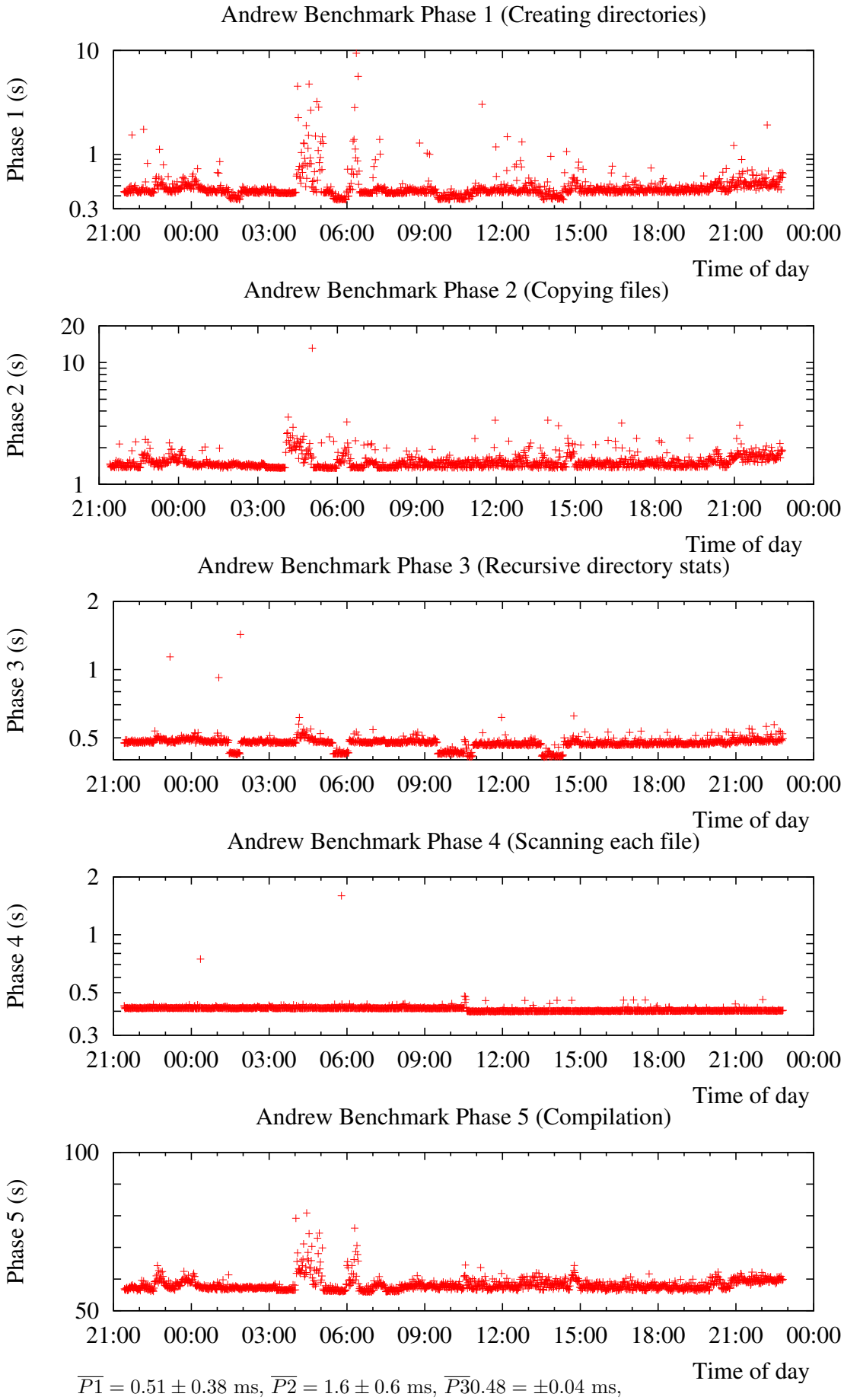
Figures 3.3 and 3.4 show the results from the modified Andrew Benchmark to the same local and a remote site. The measurements were taken over the entire 24 hour period, changing between the RTT then each of the Andrew Benchmark phases in turn. The cause of the structure in figures 3.3 Phases 1 to 3 is not known. It is possible that this may be due to the AFS servers residing behind old firewall hardware and some loss of packets is occurring causing the cache manager to resend packets. The different lines may represent different numbers of packet loss.

Figure 3.5 shows a sample of Andrew Benchmark test results from 20 tests from Manchester to AFS servers mcc.ac.gb (magenta), hep.man.ac.uk (cyan), rl.ac.uk (blue), desy.de (green) and slac.stanford.edu (red). Results shown here indicate that there should be no major impact on fairly common operation when the AFS cell that is being used is within 10 ms round trip time. The unix command `traceroute` indicates that there are around 18 hops to desy.de, 16 hops to slac.stanford.edu, 10 to rl.ac.uk, 3 to hep.man.ac.uk and 2 to mcc.ac.gb. This suggests that the nubmer



$\overline{P1} = 0.29 \pm 0.28$ ms, $\overline{P2} = 0.85 \pm 0.42$ ms, $\overline{P3}0.41 = \pm 0.18$ ms,
$\overline{P4} = 0.41 \pm 0.02$ ms, $\overline{P5} = 53 \pm 4$ ms

Figure 3.3: Andrew Benchmark to the local AFS server



$\overline{P1} = 0.51 \pm 0.38$ ms, $\overline{P2} = 1.6 \pm 0.6$ ms, $\overline{P3}0.48 = \pm 0.04$ ms,
$\overline{P4} = 0.41 \pm 0.03$ ms, $\overline{P5} = 58 \pm 2$ ms

Figure 3.4: Andrew Benchmark to a remote AFS server

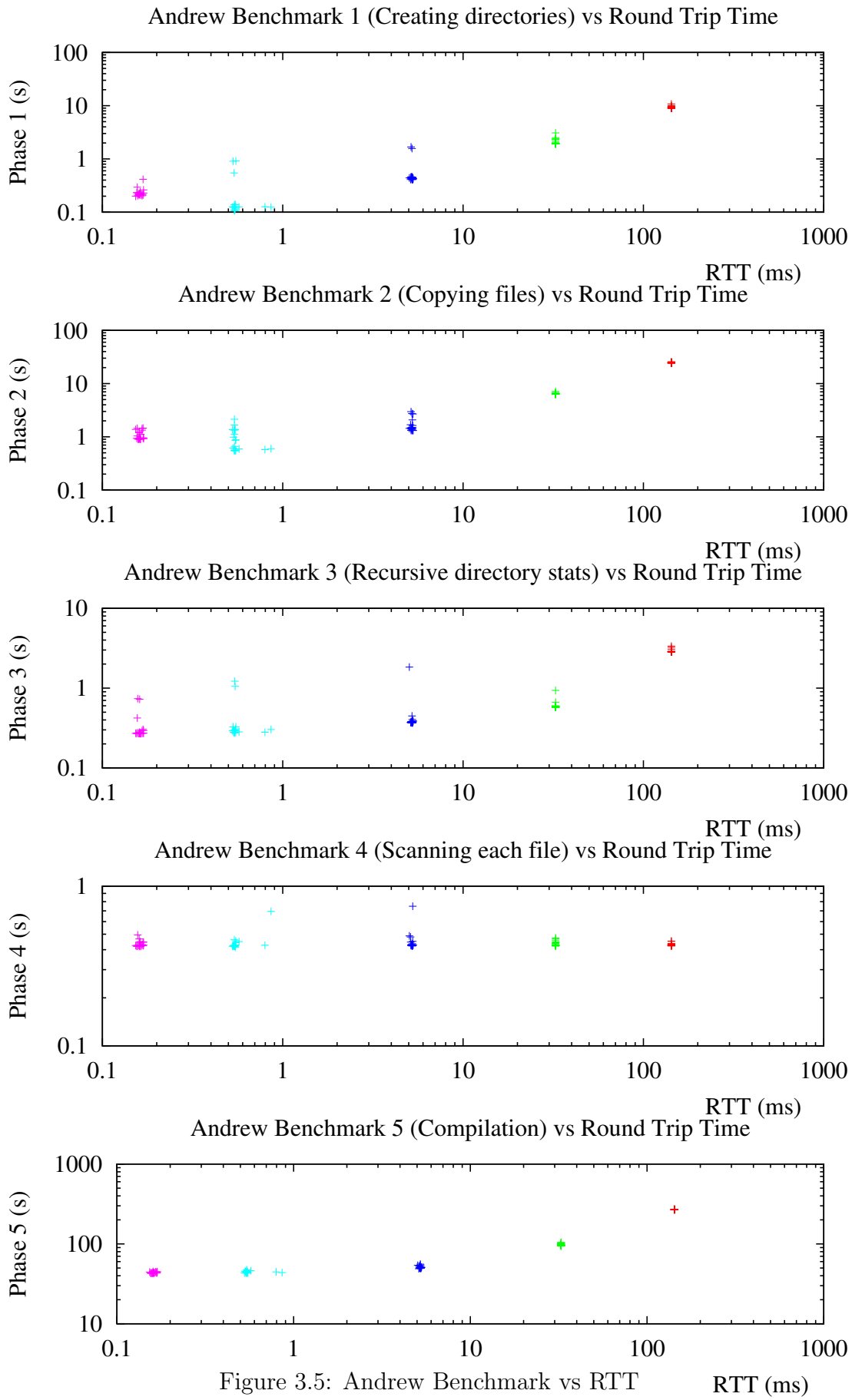


Figure 3.5: Andrew Benchmark vs RTT

Tests to differing AFS servers result in clustered points, from left to right:
mcc.ac.gb, hep.man.ac.uk, rl.ac.uk, desy.de, slac.stanford.edu

of hops has a much less significant impact on the performance than the physical distance.

Figure 3.5 also seems to suggest that remote read operations (in the Phase 4 test between Manchester and SLAC) are as efficient as local file reads. This is hard to believe; although the RTT is about 5 times faster than the test, the test still has to perform many file read operations. The likely explanation is that the cache has a valid copy of the files (having just written them). If this is indeed the case then no conclusions can be drawn from the Andrew Benchmark for operations where the cache manager would be operating at full capacity.

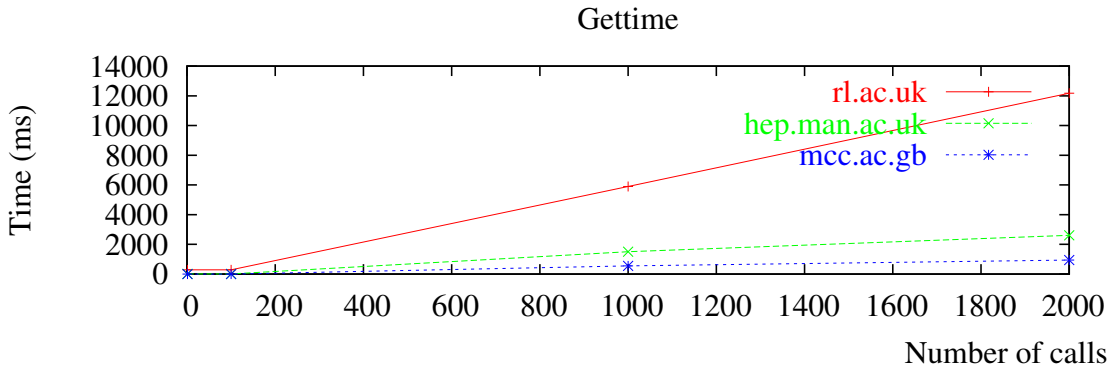


Figure 3.6: `afsfsperf` time requests

3.1.2 `afsfsperf`

The AFS filesystem performance test, `afsfsperf`, is part of the Arla [68] AFS client distribution. It is a small AFS client that performs 10 different measurements. The following tests were performed against two local cells (mcc.ac.gb and hep.man.ac.uk, with round trip time $\lesssim 1$ ms) and against a cell 250 km away (rl.ac.uk RTT ≈ 5.5 ms). It is possible to choose whether to run the tests in authenticated mode, encrypted mode, or unauthenticated. Most AFS clients run in authenticated mode and so the following measurements have been taken in that mode. Measurements with encryption switched on and with neither encryption nor authentication are shown in appendex D.

Gettime: requests the time on the remote AFS server (results are shown in figure 3.6). Each Gettime request consists of a single Rx transaction to the AFS server.

Write, Read and Remove: measure the time it takes to read, write and delete files of various sizes (results are shown in figure 3.7). The write data operation consists of a number of Rx operations. Firstly, the server is told to create the file. Then the server is told to expect data. Data is sent in many Rx datagrams and

finalised with an Rx end storage command. The read data test is similar: the file is requested, the server sends it in many Rx datagrams and the Rx context is closed. Each file is then removed with one Rx transaction.

In figure 3.7, results for mcc.ac.gb and rl.ac.uk show that both reading and writing data directly to a remote AFS server behaves as one might expect. The RTT to rl.ac.uk is about five times the RTT to mcc.ac.gb so we would expect to see a slope about five times steeper for communications to RAL than to local AFS servers. The results for hep.man.ac.uk show different behaviour, This may have something in common with the network *jitter* observed in figure 3.3.

Data in figure 3.7 Remove Data, consists of a delete commands and as such is dependent only on the network speed and file operations at the AFS server. The relatively large values to the server at RAL are only partially explained by the network latency and so are probably related to file server performance.

Create and Delete Files: test the time taken to create and delete large numbers of files (results are shown in figure 3.8). This benchmark creates many zero length files each with an Rx operation and then deletes them also with one Rx operation per file. The unusually slow response from mcc.ac.gb may be an indication that file actions have to replicated across replicated volumes.

Create and Delete Directories: test the time taken to create and delete large numbers of directories (results are shown in figure 3.9). This benchmark creates many directories then deletes them, each operation requiring a separate Rx transaction. Again, the unusually slow response from mcc.ac.gb may be an indication that file actions have to replicated across replicated volumes.

Bulk Status Operations: AFS clients allow up to 50 status requests to be sent in one message. This test measures the time taken to get the status of 5000 files, varying the number of operations in the bulk command (results are shown in figure 3.10).

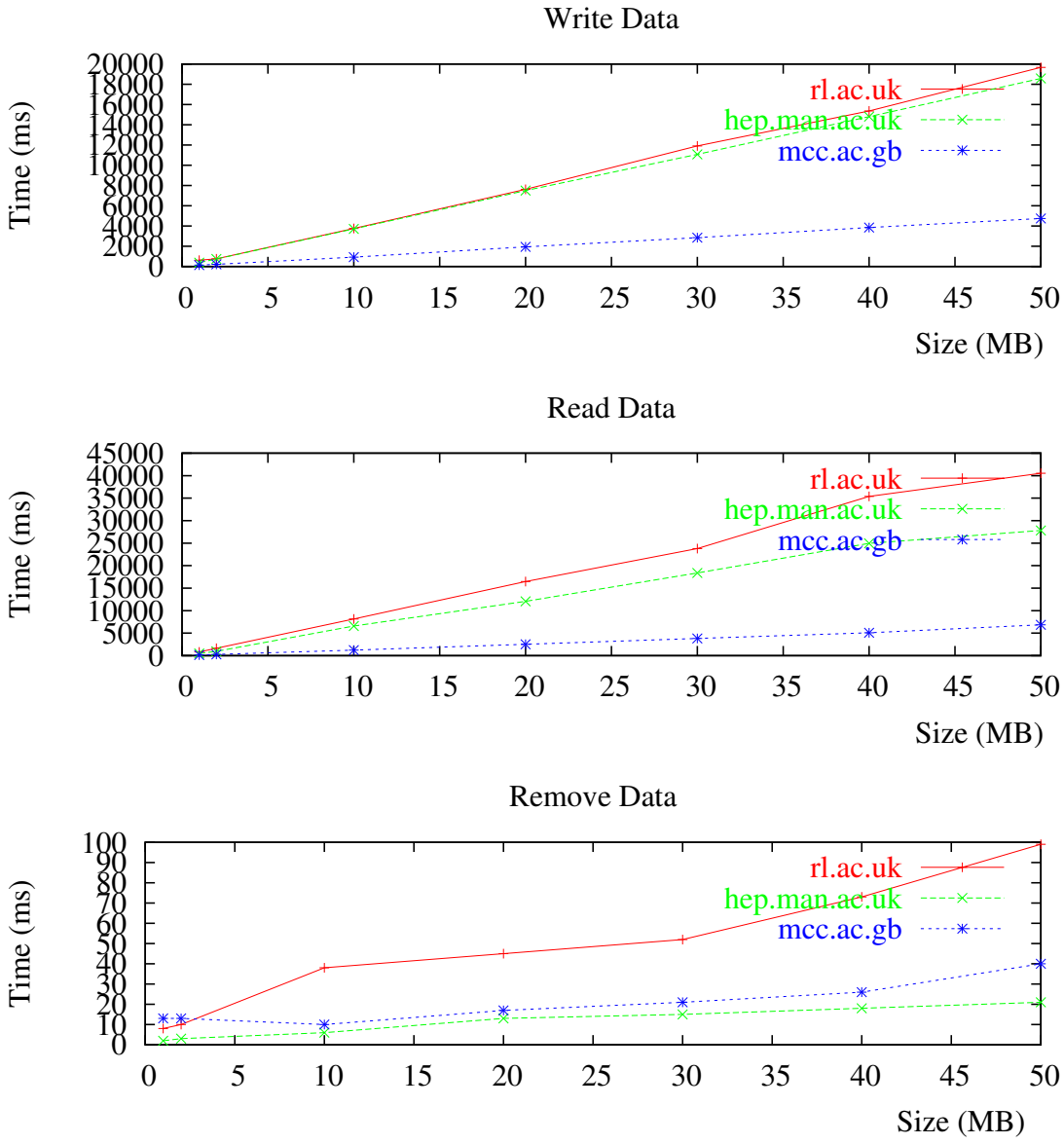


Figure 3.7: `afsfsperf` data measurements

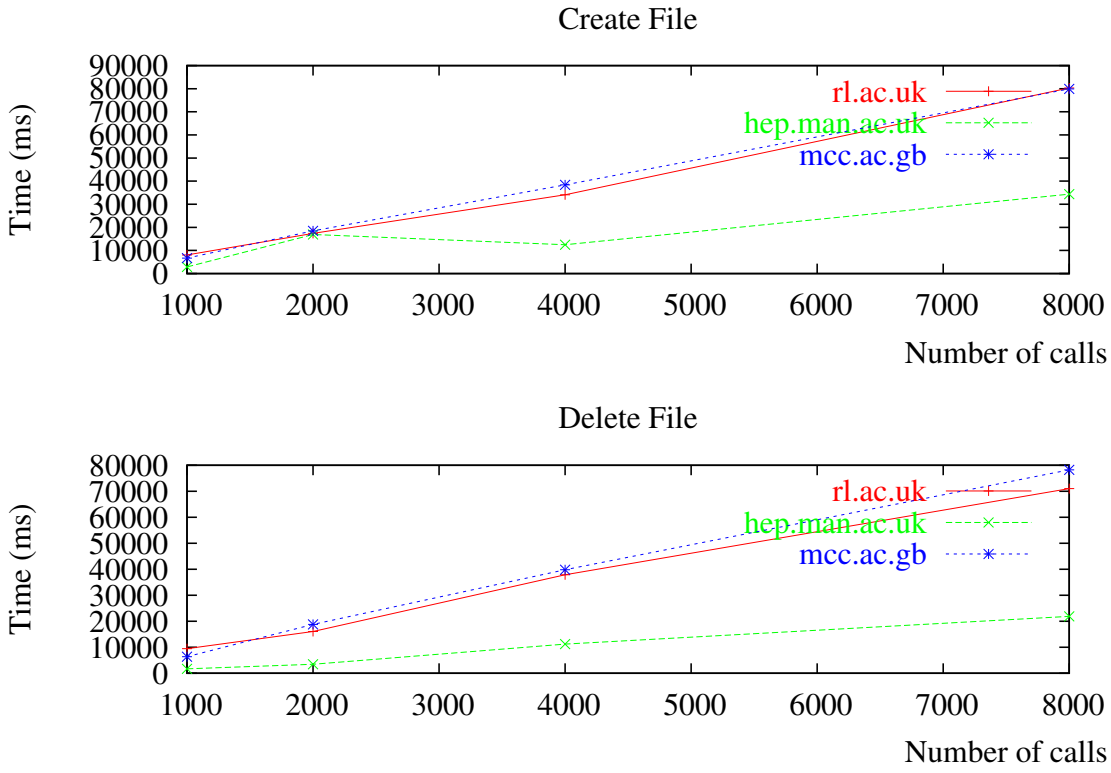


Figure 3.8: `afsfsperf` file operations

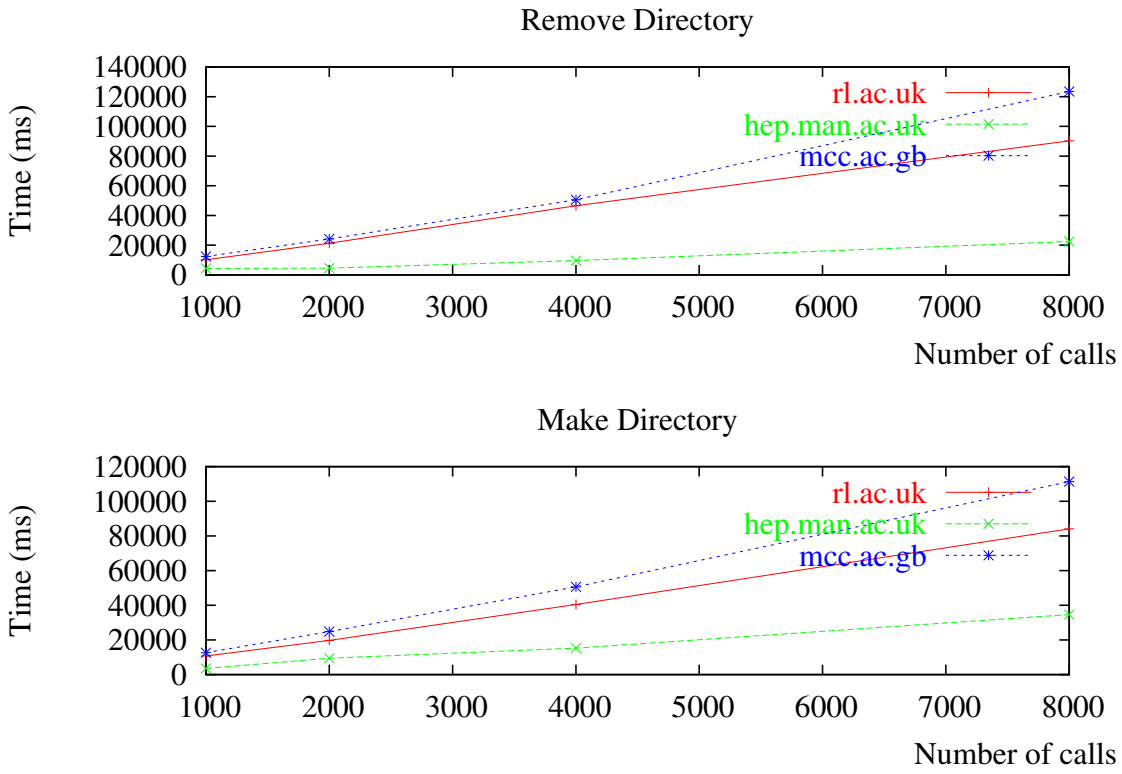


Figure 3.9: `afsfsperf` directory operations

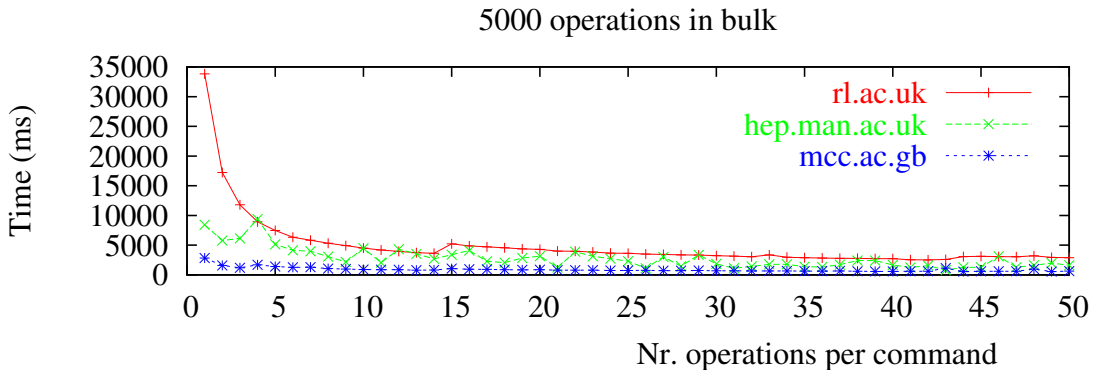


Figure 3.10: `afsfsperf` bulk operations

3.1.3 Simple Read, Write And Append Tests

A further benchmark was written for this thesis. This benchmark consists of a number of read, write and append operations carried out in recently flushed volumes. Flushing the volumes is necessary to simulate the behaviour of file operations executed on a new worker node, where there would be no cached information already.

Firstly a test directory is made. The volume is flushed. 100 files of 1 byte are created and timed. The volume is flushed. The directory is opened and all files therein are read. This provides a read time. The volume is flushed again. A third time measurement is made while the directory is reopened and each file within has one byte appended to it.

The Benchmark is run many times varying the file and append size, $1, 2, 4, 8 \ldots$ bytes. The results of these benchmarks are shown in figure 3.11 and details can be found in appendix C.

The plots in figure 3.11 show a clear independence between file size and operation time up to 1 KB. In this range the file operations are dominated by the communication of control datagrams between the AFS server and the cache manager. It is the control communication that causes the timing data for different AFS servers to have a dependency on distance from the client, hence the results for slac.stabford.edu form a curve starting at ~ 0.5 s and the local AFS servers starting at ~ 0.01 s. In the region above 1 KB, the system is experiencing the limitation of the cache size and must now wait for data communications to pass between the cache manager and the AFS server.

It can also be seen that as the file sizes increase the distances that data must travel across become less significant; thus transferring ~ 10 MB of data to (or from) slac.stanford.de is slower than the equivalent local data transfer by a factor of $\frac{1}{2}$ to $\frac{2}{3}$ compared with files of size 1 B.

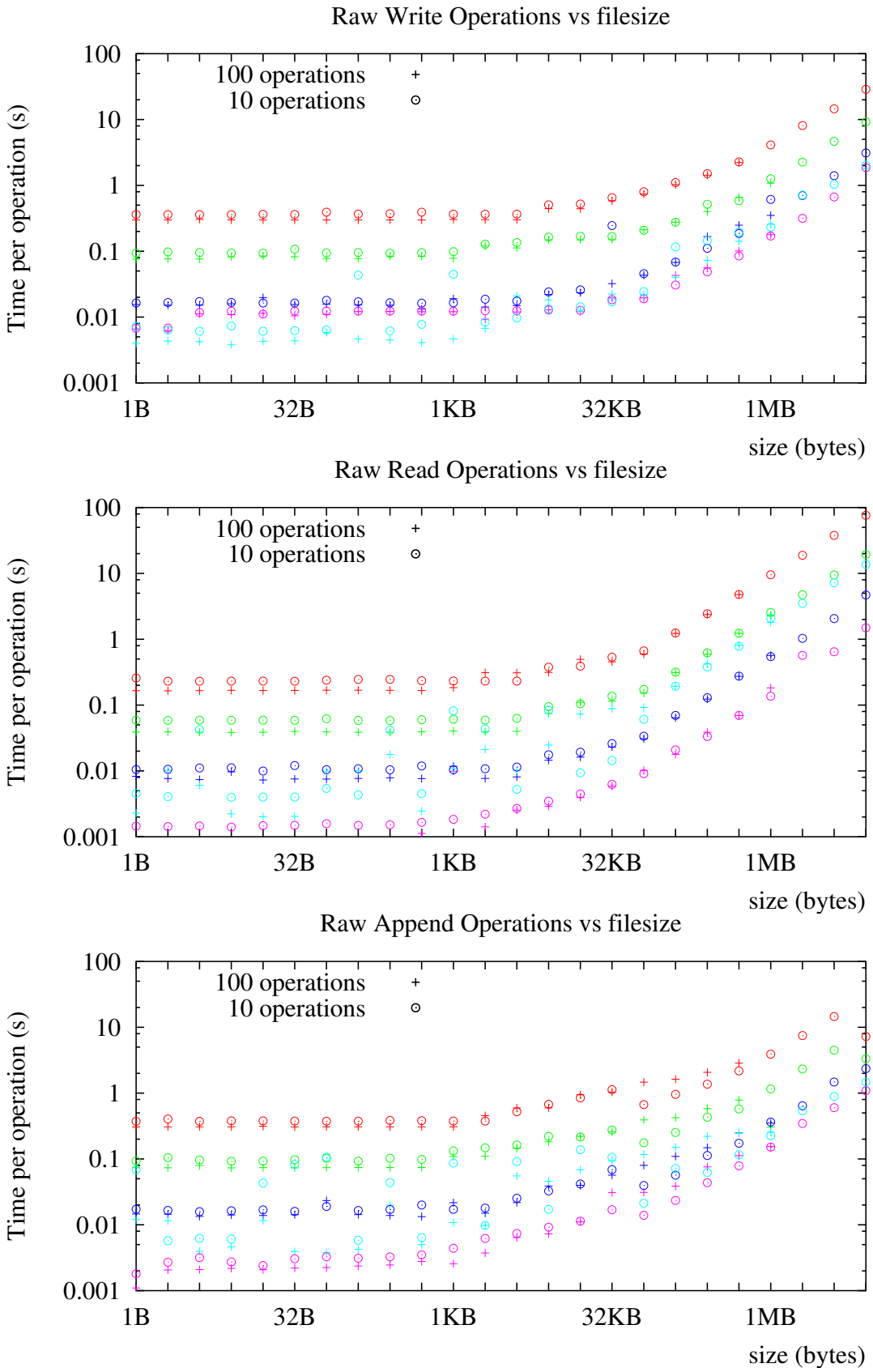


Figure 3.11: Raw data writing and reading measurements with cache management.

All measurements taken on AMD Athlon(tm) XP 2200+ 1800 GHz with a cache size of 100 KB at Manchester. slac.stanford.edu, desy.de, rl.ac.uk, mcc.ac.gb, hep.man.ac.uk

3.1.4 Comparisons

A comparison between AFS and GASS was performed by members of the Globus team, and can be found here [40]. It shows that AFS performs better in single and multiple transfers of about 1 MB or less, but for large transfers over 10 MB GASS starts to perform better.

Figures 3.12, 3.13 & 3.14 show a comparison of file transfer speed for AFS, `ssh` and GridFTP. AFS provides a better data rate for small file transfers. GridFTP and ssh fail to provide good data rates due to the authentication overhead (and extra login requirements in the case of ssh to RAL and Manchester HEP).

These comparisons suffer from the fact that one cannot use the same machines at the three sites to test server performance. AFS servers rarely allow ssh access and GridFTP access outside their local network. The plots shown here show file transfers to differing machines in the same local area network; their performance will therefore also depend on the specification of these servers. This explains the performance of ssh compared to AFS and gridFTP with the transfer of larger files.

3.2 Conclusion

Three different benchmarks and one comparison measurement were performed to a number of different AFS servers which are geographically separated.

The Andrew Benchmark §3.1.1 shows evidence that a variety of applications can operate with no major impact, provided that the AFS server is within $\sim$ 10 ms RTT. Beyond that, normal usage starts to suffer. Further conclusions cannot be drawn from the Andrew Benchmark where the capacity of the cache is exceeded.

The `afsfsperf` tests only show uncached performance. The AFS Rx operations appear to scale reasonably linearly with number of operations and file size.

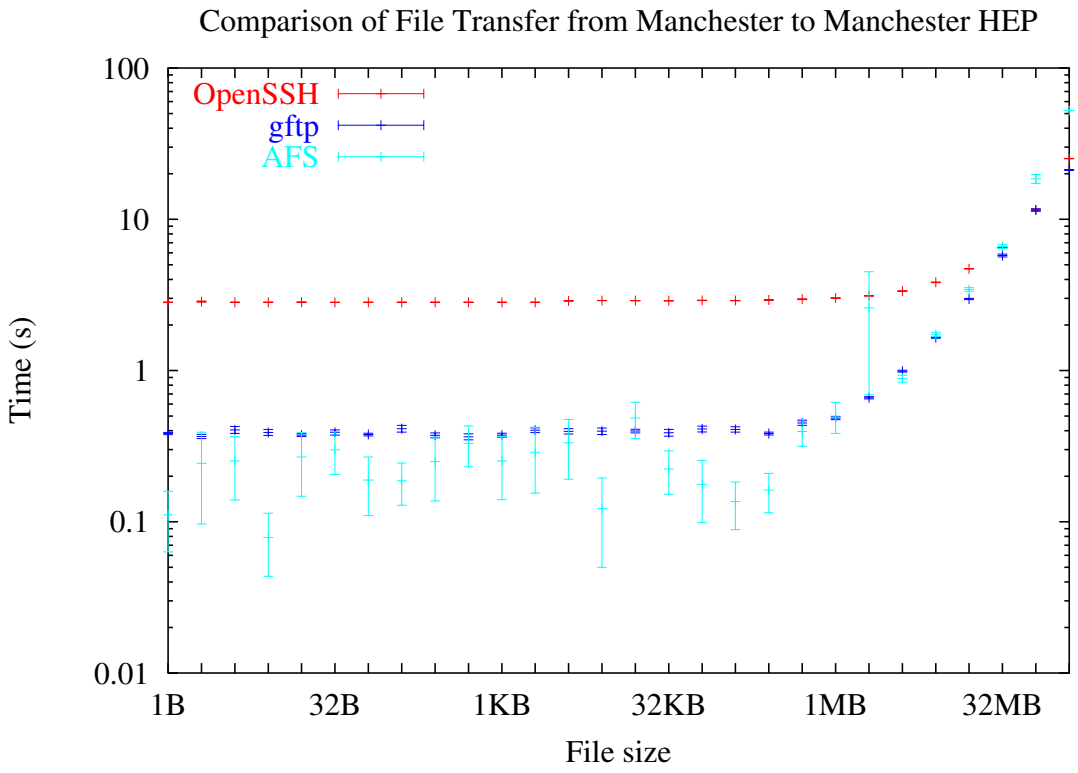


Figure 3.12: Comparison between file transfer protocols.

openSSH, gftp (a simple GridFTP client written by the Author), AFS. Measurements taken on the hep.man.ac.uk network.

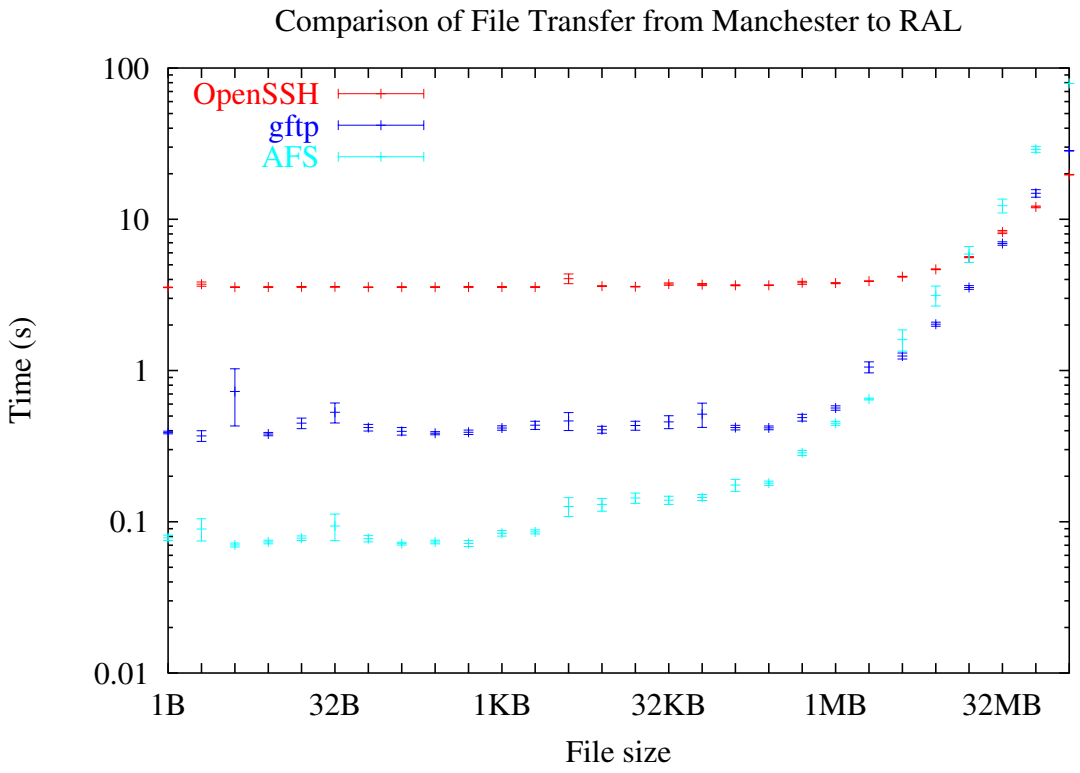


Figure 3.13: Comparison between file transfer protocols.

openSSH, gftp (a simple GridFTP client written by the Author), AFS. Measurements taken on the hep.man.ac.uk network.

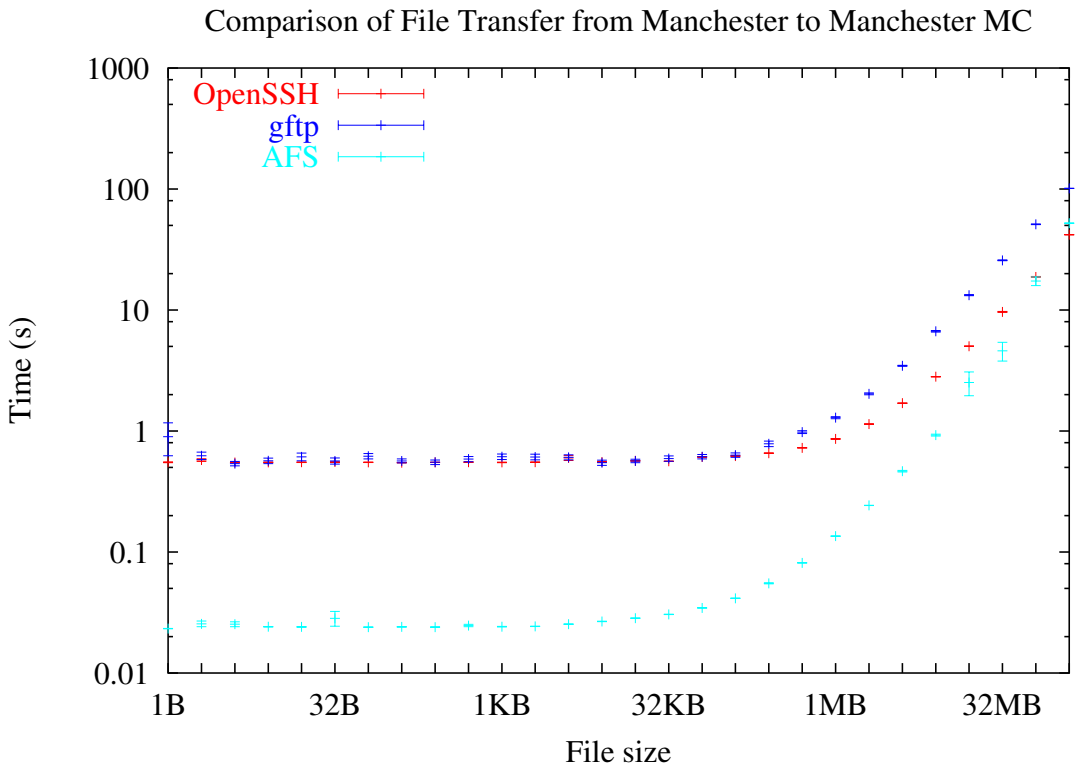


Figure 3.14: Comparison between file transfer protocols.

openSSH, gftp (a simple GridFTP client written by the Author), AFS. Measurements taken on the hep.man.ac.uk network.

From the simple write, read and append tests, it is somewhat surprising to see that at large file transfers the ratio between local and remote file access times gets smaller. Also straightforward file reading is generally believed to be faster than writing. This is true for small file transactions but appears not to be the case for larger files.

These benchmarks provide only a general hint at the level of performance of the AFS. The results in this chapter show three different aspects of the AFS' performance:

- When using the cache, resulting in biased observations for short jobs, the tests appear to run more efficiently than one might expect in a wide area network.
- When not using any cache, which only examines the underlying data protocols.
- When only dealing with file transfer, there's no illustration as to what happens when data throughput is delayed by the job's own I/O routines.

However, it is still possible to draw a couple of conclusions based on these measurements:

- When scheduling jobs that require access to AFS it is important to take into account the distance to the AFS servers (in terms of network distance e.g. as measured by the Round Trip Time).
- Jobs whose main function is to move data files across the network will start to see impact on performance when file sizes exceed $\sim$ 1 KB.

A comparison between a number of file transfer protocols measured in this thesis agrees with other external observations, that AFS performs better in a small file transfer environment.

There is really no substitute for running real analysis code in the AFS environment to see how well it behaves. This is presented in chapter 6 after the method for launching and monitoring such jobs is presented.

Chapter 4

AFS In A Globus Toolkit®2 Grid

Chapter 2 discusses the individual technologies requried to access AFS in the High Energy Physics grid environment. These are, however, independent technologies and effort is required to integrate them effectively. There are a number of approaches in which access to AFS can be engineered when running jobs on a grid.

One method would be to create a jobmanager (§2.1.1) that handles incoming requests and obtains an AFS token before handing over to the user's application. This is in fact quite complicated and would need to rely on the extensibility features in the RSL (§2.1.1). This approach is further complicated[1] by the fact that the AFS tokens are tied down to a single IP address. Jobs would have to obtain tokens on the gatekeeper at the invocation of the jobmanager then obtain another if the target execution node is different. Obtaining tokens immediately after Globus authentication is likely to delay the job submission process although this should only be significant if the network latency is large compared to the available processing power of the gatekeeper[2]. Figure 4.1 a shows the standard software stack including the grid middleware layer. Figure 4.1 b shows the middleware stack for a jobmanager that sets

[1]This might be a welcome complication if batch nodes only have access to the outside world using the gatekeeper as a NAT box.

[2]PKI is compute dominated, whereas Kerberos is network dominated.

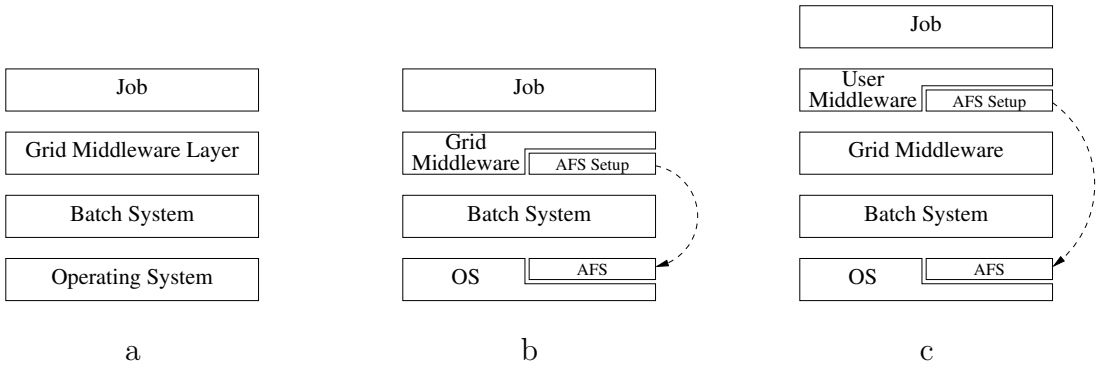


Figure 4.1: Models for providing AFS access

Figure a, shows the basic middleware stack for a Globus grid. Figure b, shows where the AFS setup might sit if it were to be handled by the jobmanager. Figure c, shows where the AFS setup sits if an extra Middleware layer is to be created before submitting a job to a grid.

AFS up on demand.

Another approach would be to provide an extra middleware layer, which sits between the Globus Toolkit®and the execution of the job, formed at the time of submission. The purpose of this layer would be to provide the job with the AFS credentials at run time. This approach removes the need for the gatekeeper to provide a special jobmanager. It also relieves the gatekeeper of the need to process any Kerberos/AFS interactions. Figure 4.1 c shows where the extra midleware layer sits in a grid if the grid middleware stack takes the form shown in figure 4.1 a.

The approach illustrated in figure 4.1 c has been implemented by the author for this thesis. The reason for this choice is that the first approach (Fig. 4.1 b) would require all sites (all grid nodes) involved to adopt the modified jobmanager[3]; whereas the second (Fig. 4.1 c) only requires the user to select the method of submission, thereby reducing the requirements imposed on each site's production systems.

The second approach is described in this chapter.

[3]Or for the LCG's Jobmanager to be changed

4.1 Prerequisites

Although the method with least deployment impact has been chosen, certain constraints are still placed on each party involved. This section covers those requirements. It is assumed that there already exists a Globus Toolkit®version 2 based grid.

4.1.1 Requirements Of The User

The user must be registered to use the grid and they must possess the means and the rights. They do this by obtaining an X.509 certificate that is trusted within that grid, joining a Virtual Organisation [10] that has rights to use that grid and using grid middleware common to that grid.

Users also need read/write access to AFS: an AFS cell that is grid enabled as described in §4.1.3 and known to the grid worker nodes.

4.1.2 Requirements Of The Grid Servers

The servers require a compute element (CE)[4], and the worker nodes (WN)[5] require AFS clients and should be visible to the network[6].

It is also assumed that secure HTTPS access is possible from the worker nodes for monitoring and tracking. This is described in the next chapter.

[4]The LCG terminology for a gatekeeper.
[5]The LCG terminology for a back-end nodes.
[6]It is possible though quite tricky to set up AFS access via a NAT.

4.1.3 Requirements Of The AFS Server

The AFS server needs to have a `gsiklog` or `gssklog` server dæmon running (see §2.3.4). Also required is a list of trusted Certificate Authorities and revocation lists along with a gridmap file. The gridmap file (similar to that which is required for Globus Toolkit®version 2 gatekeepers) is needed to map the distinguished names obtained from X.509 certificates to AFS principals on that server.

4.2 Job Submission

The job submission tool `gsub` has been created. It is a 1000 line script, written in BASH. BASH [69] was chosen because this is a common shell scripting language and in particular, is present on BaBar resources. There is no reason why this couldn't be written in Perl [70] or Python [71]. It calls the `globusrun` command with dynamically generated resource specification. The basic Globus Toolkit®version 2 client installation contains two general submission commands `globus-job-run` and `globus-job-submit`. These are similar wrappers around the `globusrun` command designed to make grid job submission user friendly.

Rather than run a remote binary or script or stage a local binary or script specified by the user, the `gsub` program dynamically creates a *wrapper* script which it stages to the remote execution host. The purpose of this script is to set up the job environment and AFS environment, before changing to the submission directory and running the user's script or binary found on that resource or in AFS.

4.2.1 Creating The Wrapper

The wrapper should only need to execute a `gsiklog/gssklog` command before executing the user's job. In practice there are a number of issues that arise.

- `gsiklog` executables are not part of the Globus core middleware distributions. They must be supplied somehow.
- Certificate Authorities in the remote execution environment may be out of date, or worse, may include a CA the user does not trust or that signs in a name space that clashes with another CA.
- AFS must be present on the worker node.
- The required AFS cells must be configured on the worker node.
- AFS tokens should only last approximately the remaining time left on the delegated proxy[7]. This is fine in theory; however, different implementations of the GSI `klog` dæmon have been observed to behave differently.
- The standard Unix environment is not generally set up by the worker node to provide any useful access to system commands and libraries.

Access To A `gsiklog` Client

The solution to the first point is to provide a `gsiklog`. This can be done a number of ways, eg, using GASS, GSIFTP, HTTP, HTTPS or AFS itself. Firewalls cause problems with GASS. GSIFTP, HTTP and HTTPS require client access to client software. Also any non-authenticated file transfer requires a security check.

AFS, is however a requirement of this grid submission already. Its only disadvantage is that, in unauthenticated mode, a file's origin and integrity cannot be guaranteed and is therefore unsafe. The gsiklog binary is provided in a world readable AFS directory. A cryptographic signature is created using a trusted certificate. Globus Toolkit®installations have SSL software and so verification is possible using tools on the workernode.

[7]The token lifetime is marked by a single byte. This has to represent lifetimes varying between 0 and 30 days.

Access To Certificate Authorities

The easiest way to have trust that the user can depend on is for the user to transfer all the CA and CRLs they trust. `gsub` gathers up all the CAs the user trusts for this particular job and bundles them as part of the script. When the script is executed on the worker node a directory is made in the remote user's home directory that contains all the CA root certificates and CRLs. The Globus environment variable `X509_USER_PROXY` is then set to point to this directory.

This method was adopted primarily to overcome misconfigurations on remote site worker nodes during the early grid testbeds. It also serves as an illustration of the depth of trust needed between the client and compute element: most grid jobs that need external authenticated communications automatically trust secondary services that the worker node 'trusts', this is not necessarily the same as what the user would trust. Using `gsub` to replicate the CA directory removes one of the many possible remote configuration errors.

AFS On The Worker Node

A test is performed by the wrapper script to see if `/afs` exists. If it doesn't and it is not checked, the user's job will fail with obscure errors. This information could in principle be obtained before hand. The command

```
grid-info-search -x -h <hostname> -p 2135 -b "Mds-Vo-name=local,o=Grid" "(objectclass=MdsFs)"
```

would produce the relevant output if the compute element were to use the default Globus Toolkit®MDS schema. However, the LCG uses an alternative schema (the GLUE schema [72]) which presents a more abstract view of storage, and as such currently cannot answer this type of question.

AFS Cell

A test is made to check that relevant cells are configured on the worker node. Practice shows that not all AFS cells advertise their existence and as such are not configured into remote worker nodes' CellServDB file[8]. Some sites also do not keep all their CellServDB files up to date.

AFS Token Timelimits

This is a significant problem and `gsub` has tackled this in a number of ways. The user's job is actually executed after a fork in the wrapper script. The child process of the fork is left to monitor the status of the parent which, after the fork, executes the user's real job. The monitor periodically checks to see if the job is still running and whether its credentials are still valid. If the credentials are about to expire the child checks to see if a new credential has arrived. If not and if configured to do so it will request a new credential from a MyProxy server. If a new credential is not then available there is little point continuing as the AFS token will expire, and file access will expire with it. In this case the child (monitoring) process kills the parent process (the user's job) and performs a few other tasks before exiting. If, however, a new proxy credential is found a new AFS token is requested and the job continues.

The Environment

The default Globus Toolkit®version 2 jobmanagers provide little or no runtime environment. When running jobs on unknown hosts with pool accounts it is impossible to set up these accounts with suitable environments in advance. The HEP community, however, has gone to great effort to define how the environment of compute resources should be set up. The wrapper script sets up, as far as possible the

[8]The CellServDB file contains a list of AFS cells, their corresponding Kerberos realms and server IP addresses.

environment sourcing files in various locations as specified by the HEPiX [73, 74]. Paths on the client machine are parsed for items in the `/afs` directory hierarchy. If present these are added to the paths on the worker node by the script.

4.3 `gsub`

Here is a run through of the basic `gsub`. The current version of `gsub` has a number of extra features tailoring it to BaBar job submission; these are discussed in chapter 6.

1. Read in command line, environment variables, or use default values to set up runtime options.
2. Check that there is a proxy certificate and that it is valid.
3. The *host clause* is constructed. This is an argument that Globus requires to tell it where to submit the job. It consists of the hostname of the the target gatekeeper, and optionally: the port it's running on, the type of jobmanager wanted and the distinguished name of the certificate used by the gatekeeper.
4. If a preconfigured wrapper script has been specified (one that was generated by a previous call) this is set up. There are two reasons for this: to be able to rerun the job at another site and to accommodate the GAnGA [75] graphical user interface which splits up job preparation and job submission.
5. If there is no pre-configured script the command line is checked for the name of an executable binary/script.
6. The AFS credentials specified are parsed; if no AFS principal/cell pairs are found these are guessed by analysing the client's system.
7. The wrapper script is then created.

8. Using Globus, the script is staged to the compute element and queued/executed by the jobmanager. There are three different methods by which this can be done.

    - As a batch submission: the job is handed to the jobmanager and left there (*cf.* `globus-job-submit`).
    - As an interactive submission, the submission waits for the job to complete relaying STDOUT and STDERR back to the client (*cf.* `globus-job-run`).
    - As semi interactive, the job is submitted in batch and the client periodically polls the compute element for the job's status and when finished it copies the STDOUT and STDERR back to the client.

4.4 Conclusion

The ease of use for the user and its behaviour on the grid testbeds has shaped the development/evolution of `gsub`. A great deal of the `gsub` code is for sanity checking: making sure that target machines are specified, environments and necessary filesystems are set up and that necessary credentials exist.

Further enhancements to `gsub` allowing it to determine necessary configuration to finally run jobs on compute elements are discussed in the next two chapters.

Chapter 5

Monitoring The Jobs And Grid Fabric

During the development of `gsub` and the first prototypes tested, the main feedback from the user community was that they found it difficult to monitor their jobs. The Globus mechanisms left them with a GASS URL which they needed to store for each submission and no way to discover this information.

`gsub` was designed to look somewhat like `qsub`[1] so tools like `qstat` and `qdel` were requested.

This chapter discusses how these tools were created, the extra requirements on the `gsub` command and how a browser interface was developed.

5.1 How To Monitor gsubbed Jobs

Jobs submitted to a Globus jobmanager directly are known only to the compute element and the submitter.

[1] `qsub` is the job submission command to the batch system PBS at Manchester HEP

There is no way to list the state of jobs queued or executing on a compute element reliably as this information is stored in the (private) *gass_cache* belonging to the UID of the jobmanager process. Hence there is no ability to see what is queued on each machine in a uniform way.

There is also a problem in that the user must store the GASS URL each time they submit a job. This is easy to do by scripting the Globus submission and storing the URL somewhere on the filesystem.

The community may want to know the state of all nodes on their grid. Are jobs failing (due to grid/queue problems)? Are the queues full? etc.

To address these issues for the BaBarGrid community a CGI script[2] hosted on a GridSite server (see §2.3.6) for GSI authentication purposes, was used to store job submission records and running statuses. At the point of grid job submission the GridSite/CGI script is informed. In response, it creates an entry and unique ID which it returns. `gsub` then weaves this ID into the wrapper script (§4.2.1) which it created. After the job is sucessfully submitted `gsub` sends the GASS URL as well as some other information to the GridSite. Both these calls use `curl` [59] with the user's proxy to authenticate and send state using HTTPS GET.

The `curl` executable is compiled and is available in world readable AFS in the same way the gssklog is (§4.2.1). At four different points, the wrapper script sends state information to the CGI script on the GridSite: when it starts (i.e. just after the batch system executes it), when the environment is successfully set up and the relevant tokens have been retrieved (i.e. just before the real job starts), when the job is finished (naturally or when a SIGTERM is received or the GSI proxy certificate is nearly expired), and each time the jobs GSI credentials have been discovered to be new.

[2]Referred to as Alibaba in publications.

5.2 CGI Program And Monitor Web Interface

The CGI Program, ($\sim$ 1000 lines of perl), responds to the following actions: query, status, submit, confirmed, started, running, finished, update, getstat, retrieve, map, and no action.

The actions: submit, confirmed, started, running and finished are designed for the `gsub` and wrapper script to upload state. When a successful message of this form is received a snippet of XML is saved to a state file representing that job. The submit action creates the file. The ID presented in subsequent communications allows the GridSite CGI script to associate the update with that job's existing state file.

The query action lists the current state and timings of known jobs at a particular site in human readable HTML (designed for browser interaction). If the action query is made using HTTPS with a recognised user certificate, extra details of the jobs submitted using that credential are made available.

The status action is similar to the query action except that the response is in XML. This is designed for the `gstat` command.

The *getstat* and *retrieve* actions allow an authenticated request with an associated proxy on the web server to ask the webserver to retrieve the current GRAM state or output of the job respectively.

The map action creates a PNG graphic describing the state of all the sites and jobs in that grid.

If the request URL has any other query string the default page is returned which displays the PNG graphic.

5.3 Further Tools

Three command line tools were developed (also written in BASH, aproximatly 100-200 lines), two to mirror `qstat` and `qdel` and one to provide a way to pass new credentials to running jobs.

5.3.1 `gstat`

The status of all jobs or just the user's jobs at all sites or at just one can be listed using `gstat`. This script uses a GSI proxy and curl to download jobs' statuses from the webserver using the 'status' action as described above. The resulting XML is parsed and a list of job statuses is returned along with each GRAM URI and ID.

The purpose of `gstat` is to provide a simple was to query what jobs are running and what jobs have finished, etc. without having to remember Job IDs and Globus URIs.

5.3.2 `gdel`

This command takes either a Globus (GASS) URI, an ID or a "-a" flag. It sends the command to delete and clean-up the job to the appropriate gatekeeper(s) (after assertaining the Globus URI if not given).

Using this script it is possible to further control running jobs without having to have saved Job ID's and/or Globus URIs.

5.3.3 `gnew`

This command will delegate a new proxy to a job based on a GRAM URL or ID. If neither are given it will determine which jobs are still active and delegate proxies

to all of them.

This is a virtually essential component for jobs that require AFS tokens for longer than the validity of the delegated proxy. An alternative to this is to use the inbuilt MyProxy functionality of `gsub`. These two methods fro renewing proxies (one push mode one pull) allow the `gsub` shepherd process to obtain newer AFS tokens as required. This is not only useful in an AFS enabled Grid but also solves long running issued to do with AFS token renewal in normal batch environments.

5.4 Conclusion

By using a GridSite server, jobs' statuses can be uploaded by the jobs themselves. A series of tools can then be used to manage running jobs in an easy and universal way. Metadata describing each job, can be recorded and only revealed by authorised queries to the webserver.

The overall status of the grid can also be monitored, queue timings can be analysed. Failed job submissions, worker node setups can be highlighted.

With four commands and a web browser for good measure, the scientist is able to submit jobs to a large number of sites, monitor them and retrieve them in a similar fashion to how s/he would run them on local resources.

Chapter 6

BaBar Analyses Using AFS In The Grid Environment

This chapter includes a paper which highlights how AFS is used in the BaBar-Grid. Section 6.2 presents some new measurements reflecting the performance of a straightforward BaBar analysis using the technologies already discussed.

6.1 A Heterogeneous Grid-based Job Submission System Used By The BaBar Experiment.

The author of this thesis presented the paper in this section as a poster and full conference paper at the UK eScience Allhands meeting September 2004 and the CHEP conference September 2004. A list of co-authors and be found on the original papers [76, 77]. A few modifications have been made to the text reflecting recent changes to the BaBar Computing Model.

6.1.1 Abstract

The BaBar experiment has accumulated many terabytes of data on particle physics reactions, accessed by a community of hundreds of users. Typical analysis tasks are C++ programs, individually written by the user, using shared templates and libraries. The resources have outgrown a single platform and a distributed computing model is needed. Grid technologies provide the natural toolset. However, in contrast to the LHC experiments, BaBar has an existing user community with an existing non-grid usage pattern, and providing users with an acceptable evolution presents a challenge.

The 'Alibaba' system, developed as part of the UK GridPP project, provides the user with a familiar command line environment. It draws on the existing global file systems employed and understood by the current user base. The main difference is that they submit jobs with a `gsub` command that looks and feels like the familiar `qsub`. However it enables them to submit jobs to computer systems at different institutions, with minimal requirements on the remote sites. Web based job monitoring is also provided. The problems and features (the input and output sandboxes, authentication, data location) and their solutions are described.

6.1.2 Introduction

BaBar [78] is a B Factory particle physics experiment at the Stanford Linear Accelerator Center. Data collected here are processed at four different Tier A computing centres round the world: SLAC (USA), CCIN2P3 (France), RAL (UK), Karlsruhe (Germany).

Data is distributed to and research takes place at several institutes in the UK, all with designated BaBar computing resources: the Rutherford Appleton Laboratory and the universities of Birmingham, Bristol, Brunel, Edinburgh, Liverpool, Manch-

ester and Imperial College, Queen Mary and Royal Holloway, University of London. With this number of sites a grid is the obvious way to manage data analysis.

The ideal view of job submission to a grid in this context is one where the user prepares an analysis code and small control deck and places this in an input sandbox. This is copied to the remote computer or file store. The code is executed reading from the input and writing its results to the output sandbox. The output sandbox is then copied back to the user.

BaBar is a real particle physics experiment, involving several hundred physicist users, that has accumulated many millions of events, filling many terabytes of storage. This simple model does not match the actual analysis of BaBar data in several respects. This is partly due to the simplicity of the model and its inadequacy to describe real world computing use patterns. It is also partly due to the history of BaBar, as its pattern of working evolved in a pre-grid, central computing environment.

Firstly, a 'job', as in a piece of work, will be split into hundreds (or even thousands) of 'jobs', in the batch system sense of the term. On input a BaBar job needs to access the large event data files, which should be mounted locally to the platform the job is running on. Much of 'analysis' is filtering: an event is read, some (not many) calculations are done, some histograms may be updated and ntuples filled, and the processing moves on. It also needs to access many runtime files containing control parameters and processing instructions. This control has been implemented using tcl, and the files are known as the "*.tcl files". .tcl files source more .tcl files, and the programs need other files (e.g. efficiency tables), dependencies quickly get out of hand. A general BaBar user prepares their job in a working directory specifically set up such that all these files are available directly or as pointers within the local file system, and the job expects to run in that directory.

One proposed solution for the job's input requirements is to roll up everything

that might be needed in the 'input sandbox' and send it: `dump` [79]. This will produce a massive file and you can never be sure you have sent all the files that the job might require. Another is to require all the BaBar environment files at the remote site, but that restricts the number of sites available to the user.

The job produces an output log - but this is usually of little interest[1], unless the job fails for some reason. The output that matters is in the histograms, which are written out to a binary file during and after the processing. To send these to the user by email is only sensible for small files and the BaBar outputs can be quite large (if it includes ntuples for further analysis.) On the other hand one could expect the user to retrieve the output themselves but this requires the user or their agent knowing where the job actually ran. Another possibility is for the job to push the output back itself, requiring write access (even a password) to the client's machine.

6.1.3 UK BaBarGrid Overview

The UK BaBarGrid is data centric, meaning that it takes less effort to move execution to the data-sets than the other way round. To this end the scientist uses `skimData` [80] to query a metadata catalogue storing event records and their location to find data. A web based interface to skimData also exists [81].

Lately the BaBar experiment has moved to the Computing Model 2 (CM2) [82]. `skimData` has been replaced with a new command, `BbkDatasetTcl` [83].

Using information about the data's whereabouts the user would normally be left to submit jobs and retrieve results at various sites manually. Alibaba allows users to submit their jobs to the compute nodes with access to nearby data and retrieve the output automatically as if the scientist was only dealing with local compute and data resources. Furthermore the scientist does not have to worry about software versions installed as their favourite version is transparently mounted on the remote

[1]Standard output and error streams can still be retrieved in the usual mannor.

resource.

Existing Grid Infrastructure

Alibaba has two main aims. The first aim is to enable scientists to access resources with minimal impact to their normal mode of operation. The secondly is to make the life of the system administrator as easy as possible. Alibaba uses existing core middleware installations and carries with it any necessary baggage i.e. it provides middleware that cannot be assumed to have been installed on the execution nodes.

6.1.4 The BaBar Virtual Organisation

Authority to run jobs on BaBar computational resources in the UK is determined either by the local system administrator or by membership to the BaBar Virtual Organisation. Membership to this VO is controlled by people who pass the same criteria which govern access to the BaBar Computing resources at Stanford. This is achieved by placing a file with details of the user's X.509 identity certificate in the user's home directory which is in the slac.stanford.edu AFS realm. A script runs regularly to pick up these files and check that the user is also in a specific AFS group (for which the user was required to sign conditions of use). The script transfers this information to an X.500 directory which can be accessed via the LDAP.

6.1.5 Globus Toolkit Version 2.x

Alibaba job submission makes use of GRAM and GSI as provided by the Globus Toolkit®version 2. Many of the sites involved have already got a Globus gatekeeper running and trusted root certificates and user lists.

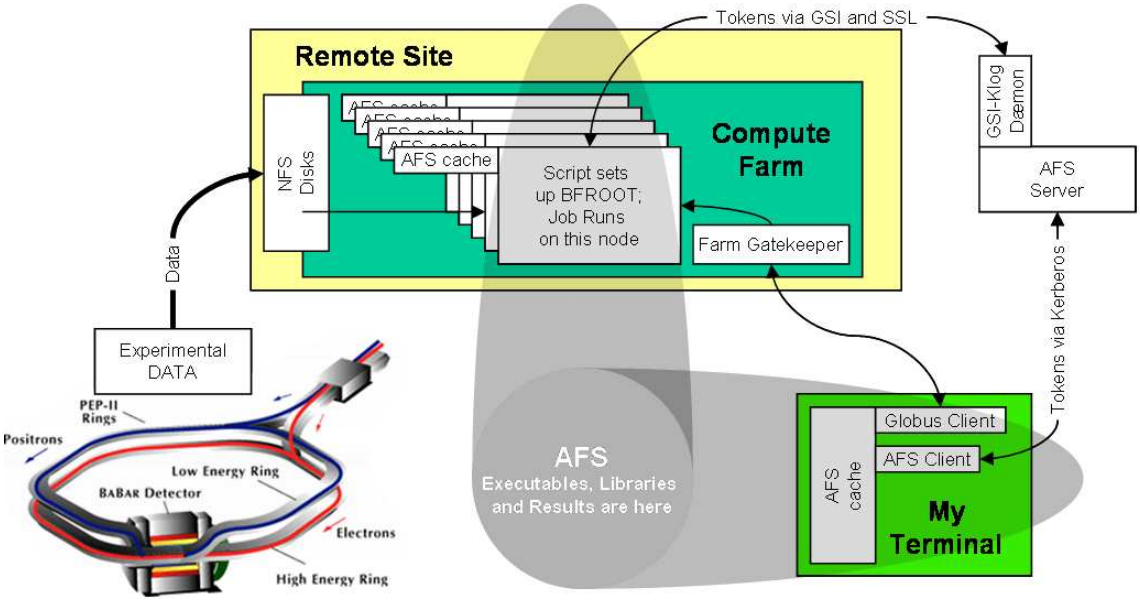


Figure 6.1: The Alibaba environment

6.1.6 EDG/GridPP Mapfile Control

To configure the BaBar Farms to authorise their use by these people and their jobs, all that is required is for the Farm's system administrator to add a script to query this database via LDAP and adjust the Globus grid-mapfile accordingly. One such script is mk-gridmapfile created for EDG [84] and LCG1 [85].

6.1.7 Alibaba Middleware

The Alibaba solution is to use AFS in place of the sandboxes. `gsub`, as described in chapter 5, effects this and hides most of the grid details. This saves the user the bother of having to collate all the necessary input beforehand and retrieve the data afterwards. Figure 6.1 illustrates the AFS solution.

A wrapper script is created automatically and staged using `globus-job-submit`. When the script starts on the remote resource it prepares and maintains the jobs

execution environment. It then forks a monitoring process, executes and becomes the job itself.

Access to the resource is based on VO management discussed above but only at the gatekeepers' end. The wrapper script executes `gssklog` [50][2] to the home AFS cell and to others if specified. `gssklog` is a self-contained binary that obtains an AFS token over SSL based on the authorisation of a grid certificate so an AFS password is not required. This executable is available to the job wrapper in a publicly accessible AFS directory and its integrity is checked by the evaluation of a cryptographic signature.

With an AFS token the job can then descend into the working directory (in the AFS). Input files can be read and output files can be written just as if running locally.

`gsub` is run (just like submitting to a familiar batch system) as follows:

```
gsub [options] command/script arguments
```

6.1.8 `gsub` On The Client Machine

1. It checks the executables and grid credentials etc.;
2. gets the current list of gatekeepers;
3. creates a script (to wrap the executable on remote batch node);
4. uses Globus to stage and submit the script to a queue on a local/remote machine;
5. uses curl over GSI-authenticated SSL to tell a website the status of the job.

[2]Actually it executes a `gssklog` (using the GSSAPI) and if that fails attempts to get tokens using the depricated GSI Globus protocols.

6.1.9 `gsub` On The Target Machine

The script created by gsub is staged through the Globus interface to the remote batch system and does the following:

1. it sets up a normal environment [73]. Globus by default does not provide a sensible execution environment; it must be built at run time;
2. notifies a website for monitoring;
3. gets (Process Authentication Group (PAG) separated) AFS credentials using `gssklog`. Each job runs with access to its own AFS credential so that it may be removed when the job finishes;
4. creates `$BFROOT`[3] the BaBar directory system and replicates the BaBar environment;
5. it changes to the working directory;
6. starts a shepherd process (this will look after job's grid credentials and talk to the website and MyProxy[9] server if required);
7. runs the user's executable (script or binary);
8. unlogs (removes credentials no longer required).

6.1.10 Monitoring The Job And The Grid

`gsub`-submitted jobs upload their progress information to a GridSite [53] (the grid enabled Apache webserver). Information handled this way is uploaded securely (XML files over HTTPS). Authorisation to upload information is based on the standard mutual authentication of an HTTPS server but using the GSI proxy credential

[3]BFROOT is a directory tree with a specific structure containing BaBar releases, executables, libraries and links to local data.

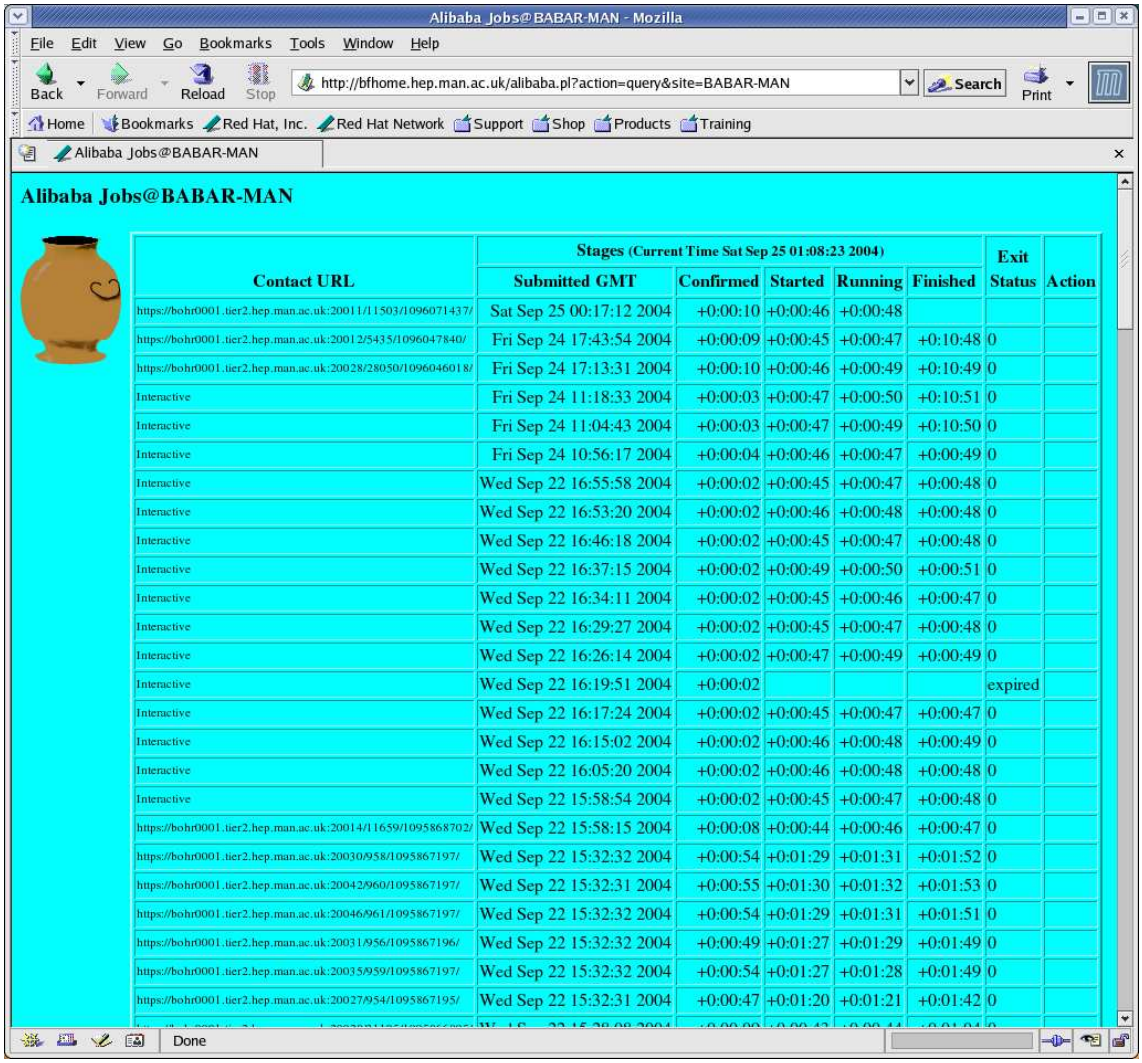
available to the running job rather than a standard X.509 certificate. Only credentials issued by the owner of the job can alter job status files on the webserver after the initial job submission.

Information about the job is available in parts to the general public via HTML (see figure 6.2). The job's owner is able to access more information by visiting the same website and authenticating using their X.509 certificate (figure 6.3). The information available to the authenticated user is:

- The times at which the job's status changed.
- The IP address of the batch node.
- The x-gram URL of the gatekeeper.
- The UID of the submitter.
- The UID as mapped on the batch node.
- The exit status of the job.
- The Globus ID of the job.
- The time remaining of the delegated proxy.

This information allows the user to orchestrate their job submissions and pick up errors as they occur. It also allows the monitoring of proxies which may need updating (either automatically via MyProxy) or manually through Globus. A suite of tools have been written to make use of this data: `gstat` - to get job statuses, `gdel` - to delete running/queued jobs and `gnew` - to push a proxy to a running job.

One further advantage of uploading the jobs' statuses to a single server is that this information can be used to present an overview of the grid. Figure 6.4 shows a status map of the grid enabled BaBar resources in the UK. The average time to execution is available and the numbers of jobs in their various stages can be seen.



Contact URL	Stages (Current Time Sat Sep 25 01:08:23 2004)					Exit Status	Action
	Submitted GMT	Confirmed	Started	Running	Finished		
https://bohr0001.tier2.hep.man.ac.uk:20011/11503/1096071437/	Sat Sep 25 00:17:12 2004	+0:00:10	+0:00:46	+0:00:48			
https://bohr0001.tier2.hep.man.ac.uk:20012/5435/1096047840/	Fri Sep 24 17:43:54 2004	+0:00:09	+0:00:45	+0:00:47	+0:10:48	0	
https://bohr0001.tier2.hep.man.ac.uk:20028/28050/1096046018/	Fri Sep 24 17:13:31 2004	+0:00:10	+0:00:46	+0:00:49	+0:10:49	0	
Interactive	Fri Sep 24 11:18:33 2004	+0:00:03	+0:00:47	+0:00:50	+0:10:51	0	
Interactive	Fri Sep 24 11:04:43 2004	+0:00:03	+0:00:47	+0:00:49	+0:10:50	0	
Interactive	Fri Sep 24 10:56:17 2004	+0:00:04	+0:00:46	+0:00:47	+0:00:49	0	
Interactive	Wed Sep 22 16:55:58 2004	+0:00:02	+0:00:45	+0:00:47	+0:00:48	0	
Interactive	Wed Sep 22 16:53:20 2004	+0:00:02	+0:00:46	+0:00:48	+0:00:48	0	
Interactive	Wed Sep 22 16:46:18 2004	+0:00:02	+0:00:45	+0:00:47	+0:00:48	0	
Interactive	Wed Sep 22 16:37:15 2004	+0:00:02	+0:00:49	+0:00:50	+0:00:51	0	
Interactive	Wed Sep 22 16:34:11 2004	+0:00:02	+0:00:45	+0:00:46	+0:00:47	0	
Interactive	Wed Sep 22 16:29:27 2004	+0:00:02	+0:00:45	+0:00:47	+0:00:48	0	
Interactive	Wed Sep 22 16:26:14 2004	+0:00:02	+0:00:47	+0:00:49	+0:00:49	0	
Interactive	Wed Sep 22 16:19:51 2004	+0:00:02				expired	
Interactive	Wed Sep 22 16:17:24 2004	+0:00:02	+0:00:45	+0:00:47	+0:00:47	0	
Interactive	Wed Sep 22 16:15:02 2004	+0:00:02	+0:00:46	+0:00:48	+0:00:49	0	
Interactive	Wed Sep 22 16:05:20 2004	+0:00:02	+0:00:46	+0:00:48	+0:00:48	0	
Interactive	Wed Sep 22 15:58:54 2004	+0:00:02	+0:00:45	+0:00:47	+0:00:48	0	
https://bohr0001.tier2.hep.man.ac.uk:20014/11659/1095868702/	Wed Sep 22 15:58:15 2004	+0:00:08	+0:00:44	+0:00:46	+0:00:47	0	
https://bohr0001.tier2.hep.man.ac.uk:20030/958/1095867197/	Wed Sep 22 15:32:32 2004	+0:00:54	+0:01:29	+0:01:31	+0:01:52	0	
https://bohr0001.tier2.hep.man.ac.uk:20042/960/1095867197/	Wed Sep 22 15:32:31 2004	+0:00:55	+0:01:30	+0:01:32	+0:01:53	0	
https://bohr0001.tier2.hep.man.ac.uk:20046/961/1095867197/	Wed Sep 22 15:32:32 2004	+0:00:54	+0:01:29	+0:01:31	+0:01:51	0	
https://bohr0001.tier2.hep.man.ac.uk:20031/956/1095867196/	Wed Sep 22 15:32:32 2004	+0:00:49	+0:01:27	+0:01:29	+0:01:49	0	
https://bohr0001.tier2.hep.man.ac.uk:20035/959/1095867197/	Wed Sep 22 15:32:32 2004	+0:00:54	+0:01:27	+0:01:28	+0:01:49	0	
https://bohr0001.tier2.hep.man.ac.uk:20027/954/1095867195/	Wed Sep 22 15:32:31 2004	+0:00:47	+0:01:20	+0:01:21	+0:01:42	0	

Figure 6.2: The Alibaba widely available information.

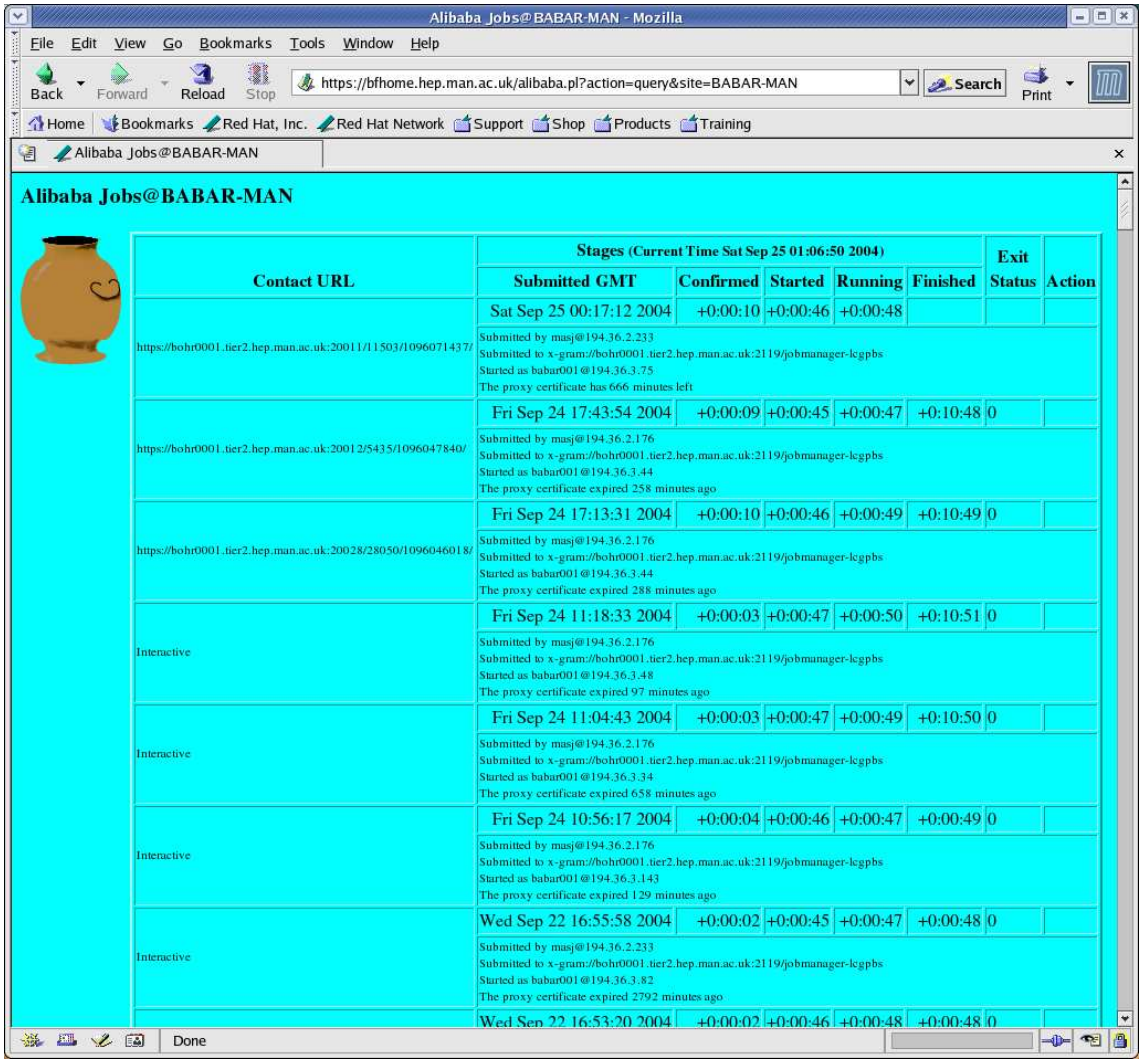


Figure 6.3: The Alibaba private information.

This information can be used by the administrators for maintenance purposes and by the scientists to help them decide where to run jobs.

6.1.11 Alibaba Via GAnGA

`gsub` has been modified to split the job preparation and submission components to allow GAnGA [75] to interface with it. A Graphical User Interface has been constructed within this framework and is shown in figure 6.5.

6.1.12 Conclusions

The Alibaba solution has the following advantages.

- There are minimal requirements at the remote host site: the gatekeeper, having Globus and some execution system, need not provide extra environment, nor deal with the extra requirements for AFS credential gathering. The worker nodes do not require any BaBar software (though the operating system has to be compatible with the binary).
- It is not required that the remote sites supply the `gssklog` software as everything needed by the middleware layer is available in world readable AFS or in the wrapper script itself.
- Remote sites do have to have an AFS (client), and to have IP communication with the outside world. Given the widespread use of AFS in HEP, this is not so problematic a requirement. Whether worker nodes at future computer centres will or should have IP access is being hotly debated at the present time: the centres are reluctant but the users are demanding it.
- These is minimal impact to the user. The user executes commands in their `workdir` directory just as with the original centralised computing model sys-

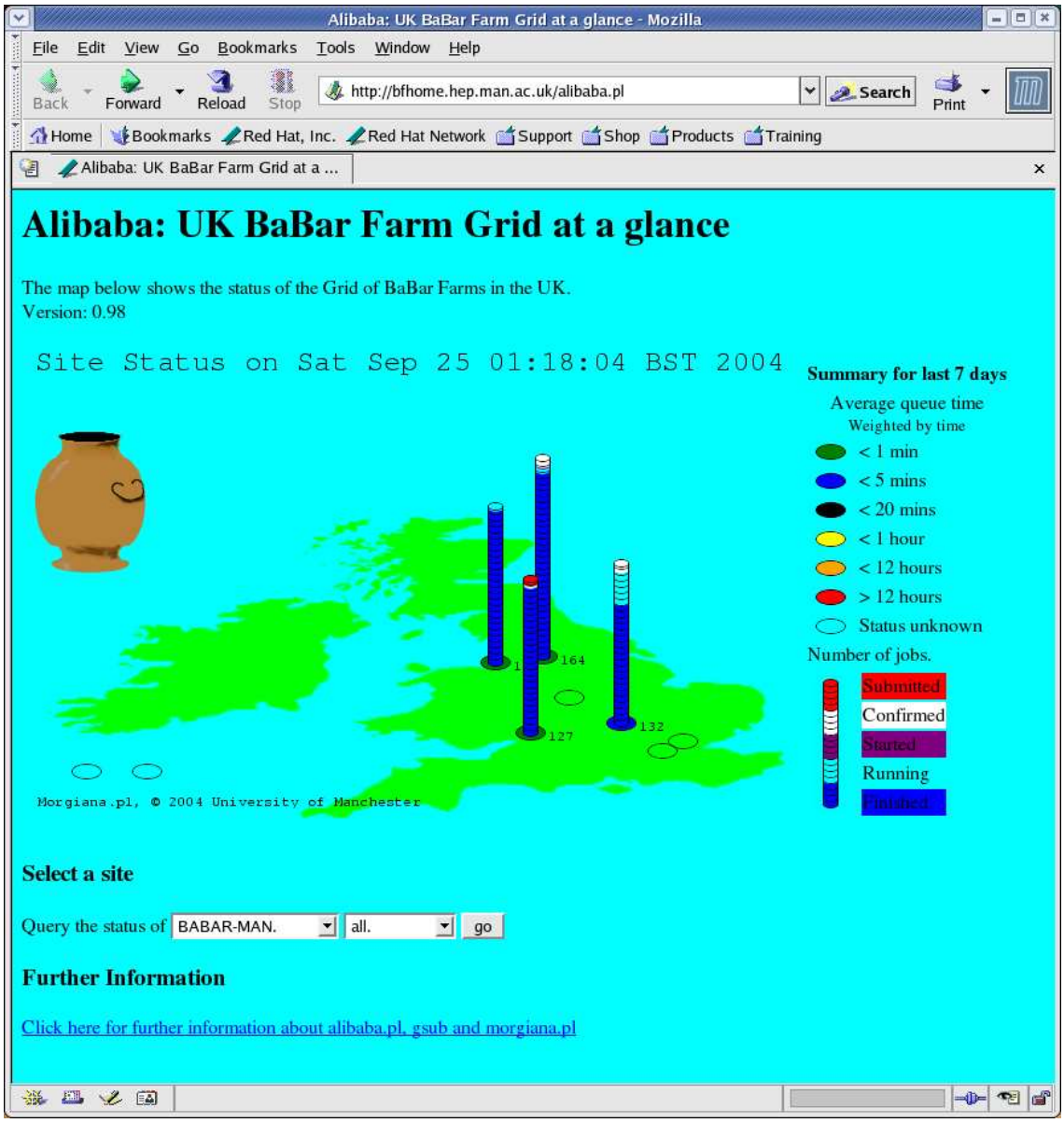


Figure 6.4: The Alibaba status map.

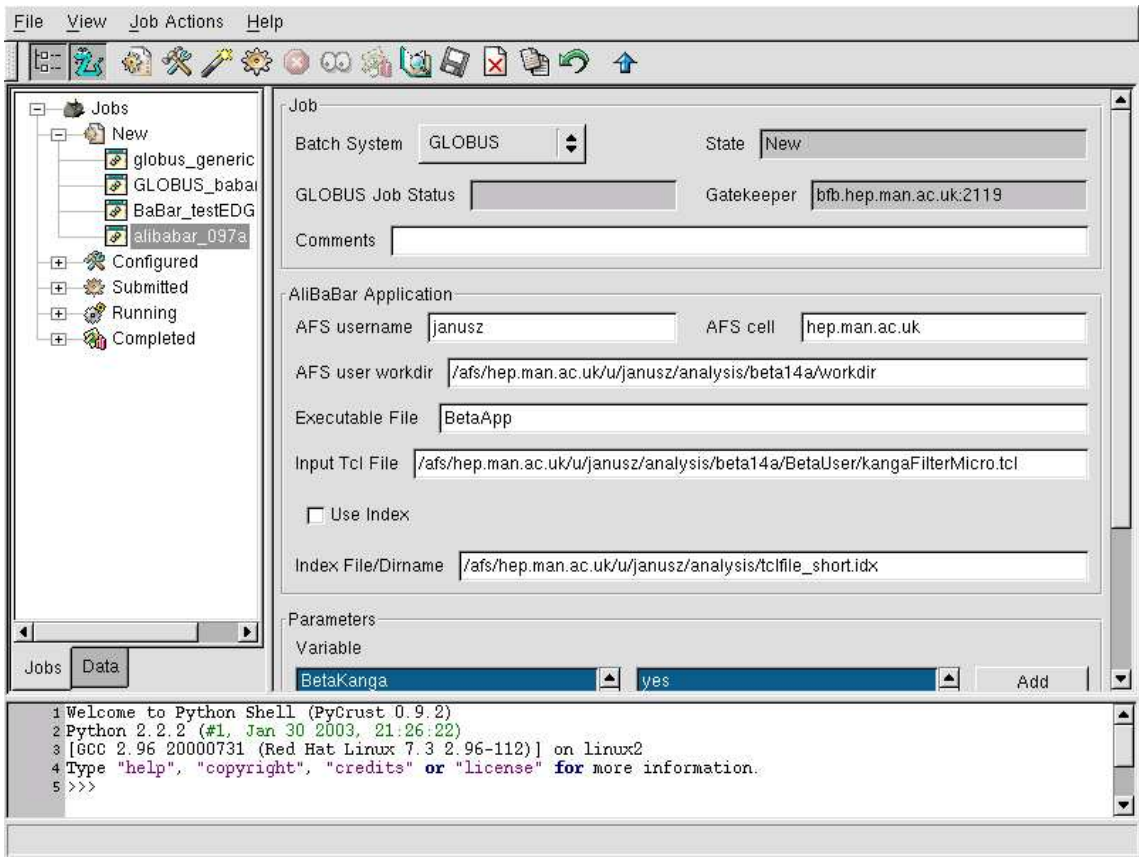


Figure 6.5: The GAnGA GUI developed to use Alibaba

tem. Whatever is available in that directory and cell is available to the remotely running job. The output histogram files appear in that directory just as they do if a job is running on the local system.

Alibaba provides a framework for easy job submission, grid monitoring and the tracking of jobs; three key requirements when moving from the desktop to a grid environment.

6.2 BaBar Code Performance In AFS

A number of tests were performed using the *BetaMiniApp* analysis from two different releases at two different sites, the Rutherford Appleton Laboratory and the University of Manchester. Both these institutes have their own AFS servers and `gsiklogd` servers.

The tests presented below aim to discover two things. Firstly, whether submitting a number of jobs to run concurrently will cause a significant bottleneck in the data transfer components of the job. The second test explores how the job itself behaves when varying the number of events analysed; from a low number of events (where the main component of the job is expected to be set up and library linking), to a large number of events where the initial set up is expected to be of low cost compared to computation and local data access).

6.2.1 Performance Based On Number Of Concurrent Jobs

For this test, the number of events to be analysed is 1000. This is a relatively low number of events to analyse, intended to exaggerate the AFS network activity of the job. Figure 6.6 shows the effect of number of separate jobs started at the same time. The tests take place on production systems and so are restricted to queuing

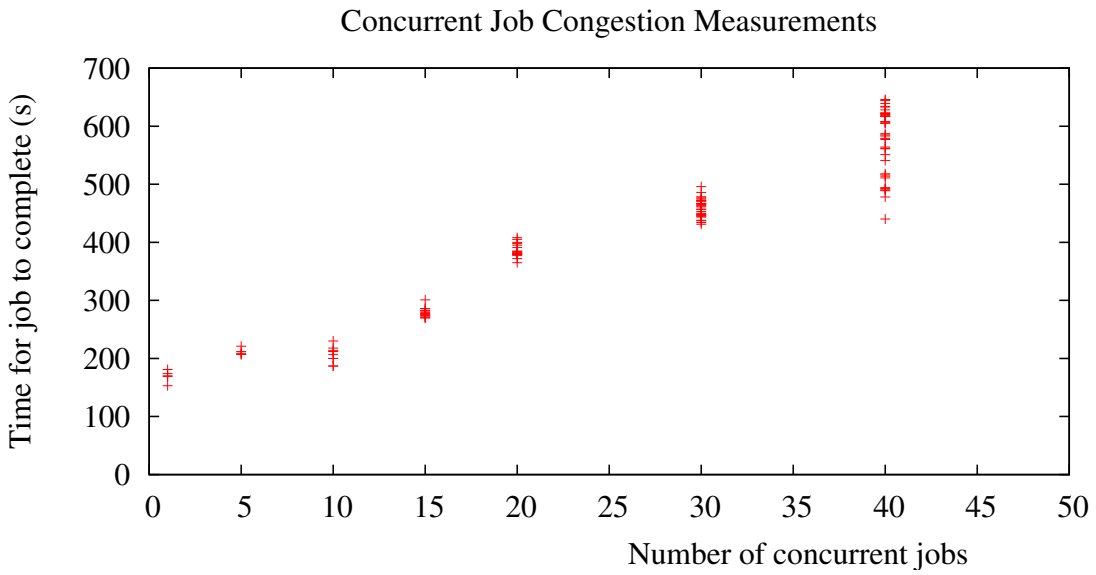


Figure 6.6: Measurements of the effect of running jobs concurrently

rules.

6.2.2 Performance Based On Number of Events

The tests in this section show the timings of small and large analyses. Jobs varying in number of analysed events from 1 to 100,000 (and later 1,000,000) were submitted and timed to both local and remote AFS enabled grid resources. The timings are recorded by a modified `gsub` command which executes the *BetaMiniApp* between two calls to `gettime` (the timing program used for the modified Andrew Benchmark).

Figure 6.7 shows the performance of two similar analysis codes running at two sites accessing similar data and executables from local servers. The performance of the code on the RAL based machine is clearly better especially when the analysed dataset is large. When the number of events is low the speed of the analysis tends towards a distinct set-up time. These set-up times are relatively similar.

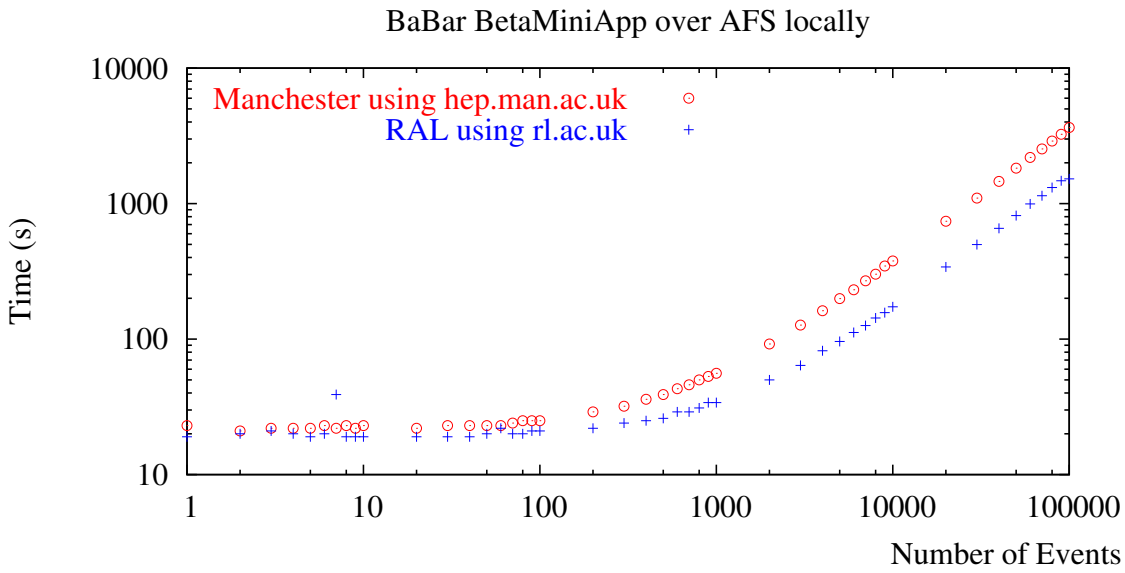


Figure 6.7: BetaMiniApp running locally.

Figure 6.8 shows the performance of the BetaMiniApp analysis attached to Manchester's AFS running remotely at RAL. The two lines represent the data points from figure 6.7. Glitches of $2-3$ minutes can be seen, exaggerated by the log scale. For this reason the same analysis was run again to the same place and is shown in figure 6.9. Closer inspection shows these glitchs (highlighted by the yellow band) correspond with a change of worker node. Each time a new worker node is assigned by the batch system (different to one previously used), a clean or expired cache is encountered and so all the libraries and executables are not available locally (in that cache) and have to be fetched.

The tests were rerun (figure 6.10 and 6.11) with a further modified `gsub`. The modifications forced all the AFS volumes for the working directory and BFROOT to be flushed before the BetaMiniApp was called. This forced the cache manager to release all AFS files in those volumes and therefore all libraries and files required by the BetaMiniApp will have been sent for at the start of its execution. This is a more likely scenario for user based jobs on a grid. A sample of each measurement

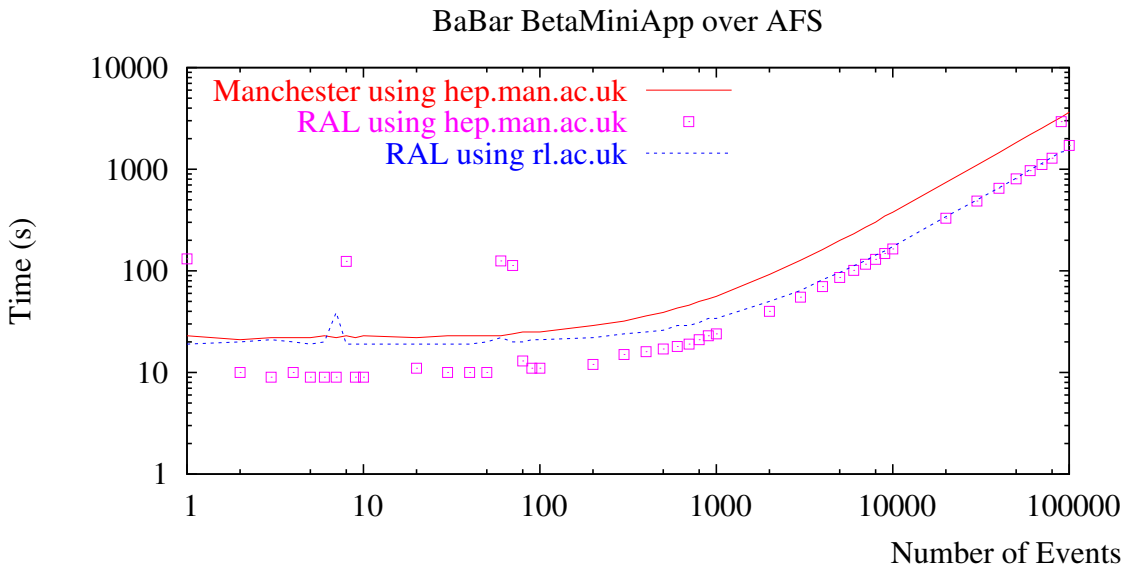


Figure 6.8: BetaMiniApp running remotely.

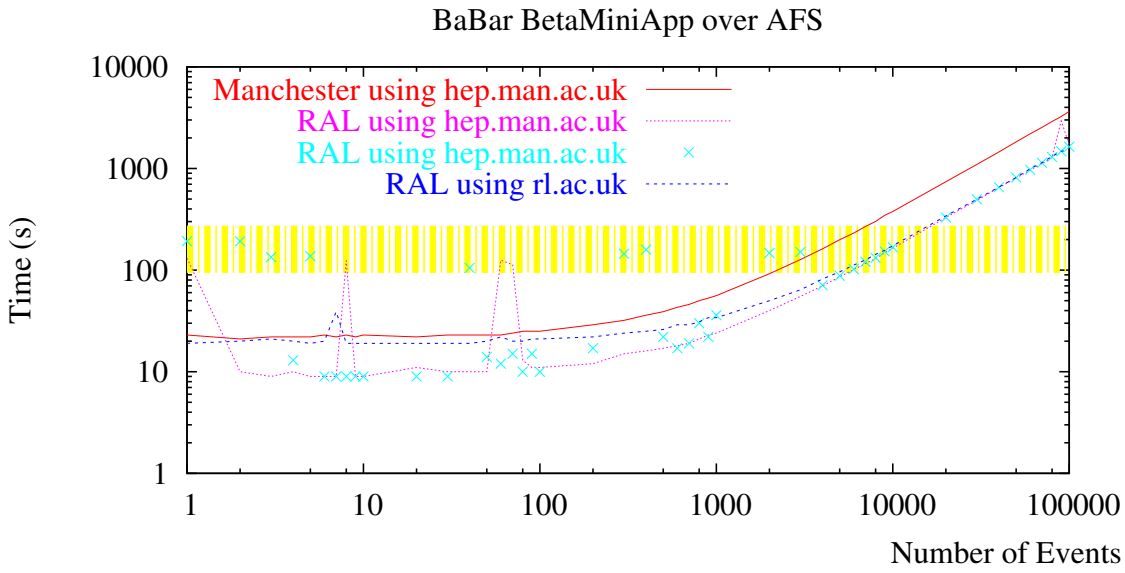


Figure 6.9: BetaMiniApp running remotely (repeat tests).

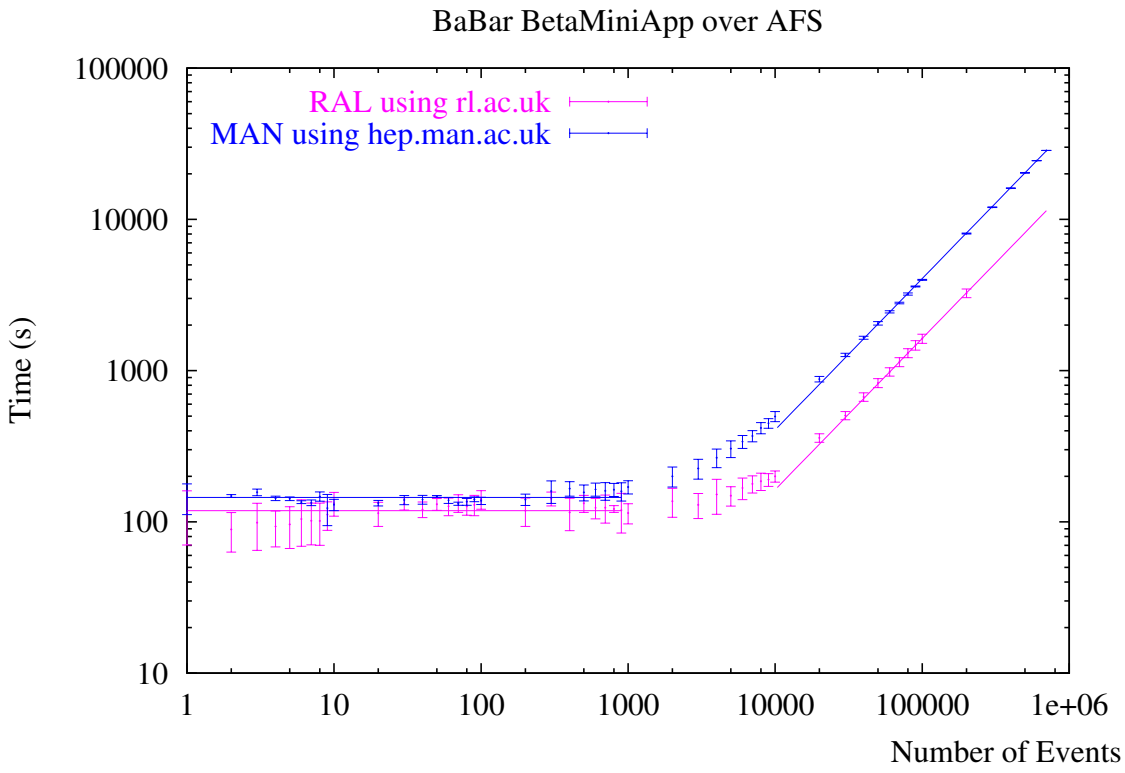


Figure 6.10: BetaMiniApp running locally with a clean cache.

was taken by running the analysis ten times in a loop in the same `gsub` command. The error bars indicate a simple standard devation of these results. These are only illustrative of the network activity due to the non-linear behaviour of networks.

Figure 6.10 shows access to local AFS file servers. Having the cache cleaned before running has a large impact on the small local jobs. Two fits were made to each dataset, one to show the linear dependence of the running time on the number of events and the other to show the constant set-up time.

Figure 6.11 shows the results obtained when running at Rutherford Appleton Laboratory using the Manchester AFS server. With the four fits from figure 6.10 superimposed.

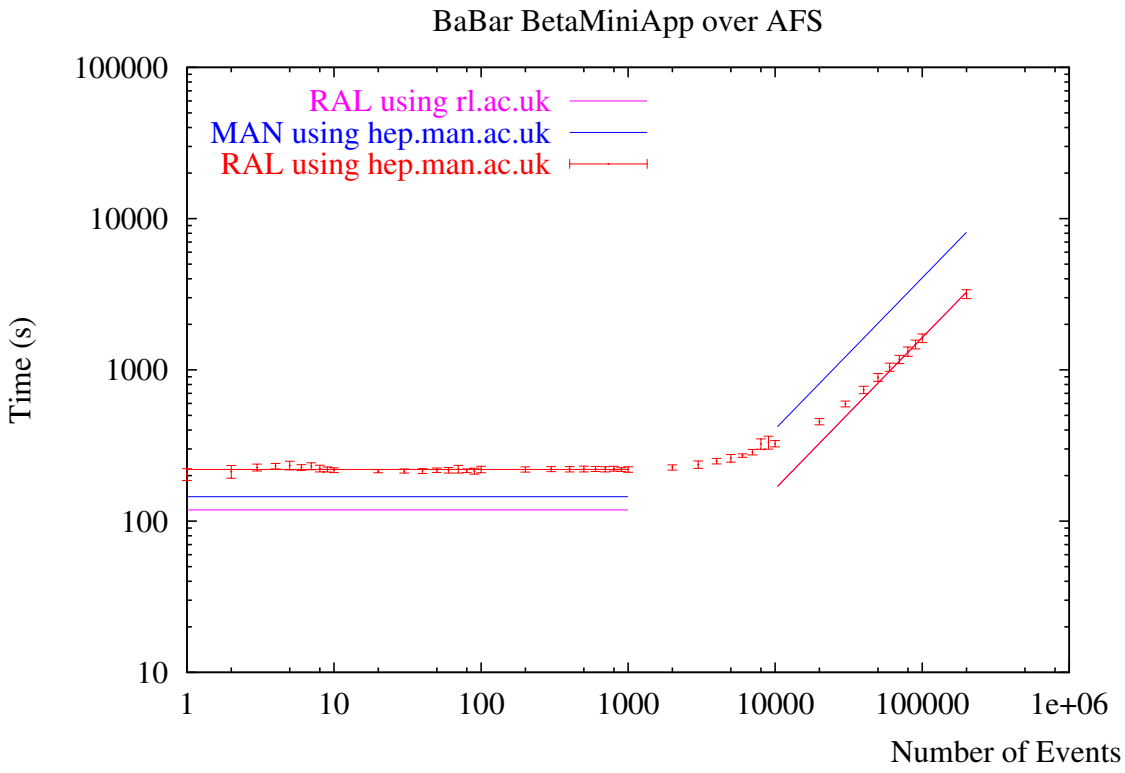


Figure 6.11: BetaMiniApp running remotely with a clean cache.

6.3 Conclusion

Section 6.1 shows the advantages of moving analyses to a grid environment and that AFS integration is naturally the way to deal with the complex BaBar analysis models.

The performance measurements obtained by running a simple CM2 framework BaBar analysis on various grid enabled resources shows that contributions to the overall speed of that analysis due to the use of AFS is not significant when taking into account the differing specifications of those resources.

There could be a short delay when multiple jobs start at the same time. It is likely that this delay occurs at the volume server. AFS provides the facility to have multiple volume servers which would reduce this bottleneck.

Chapter 7

Discussion

7.1 Real World Facing Worker Nodes

Scaling a grid that uses the AFS, to sites with a limited number of IP addresses or a draconian firewall policy is difficult. It is generally not possible to convince these sites to expose their worker nodes to the outside world. There are a few possible solutions to this: NAT, the AFS to NFS translator and Virtual Private networks.

A NAT solution has the problem that the timeouts for UDP are generally set too low. This can be overcome with care but there are many NAT products on the market with many differing specifications.

The AFS/NFS translator [86] was essentially designed to allow machines, for which there was no port of the AFS Client, to access the AFS. It does this using a single AFS client which exports the AFS file-system using NFS, translating the AFS ACLs into user/group file attributes as far as possible.

7.2 AFS Jobs In A Batch Environment

Renewal of tokens in a batch environment has been a problem for many years. Allowing the job to take with it a ticket granting ticket may be one solution but the TGT will also expire some time. Infinite length or 30 day tokens are another solution but there is no convenient way to revoke this authority once granted.

The MyProxy method for renewing GSI credentials and the subsequent request to a `gsiklog` server as presented in this thesis provides an easy, user-revocable, method for this.

7.3 AFS Is Not Grid Enough

During the evolution of this thesis there was a heated discussion on the SLAC hypernews complaining that AFS was not grid enough [87]. AFS is distributed (§2.2). AFS does deal with differing administrative domains (§2.2.1).

7.4 AFS Demand On Workers

One issue that may result from worker nodes using AFS is that demand for AFS may result in a bottleneck in the network routers to the execution farm. However, this can be weighed against the latency of file access from the wide area network. If AFS is a feasible file system for a local farm then, provided it is suitable for one job on the wide area network, it should scale up to multiple user usage.

7.5 AFS Demand On Servers

If the scientific research scales to such an extent that all grid sites are accessing AFS in one cell, then clearly this may result in denial of service. There are two points here.

Firstly, the aim for a 'ubiquitous grid' means that demands on any one server is reduced by the availability of other servers in possibly differing administrative domains. If AFS is chosen as a grid file system then the number of users accessing this grid will be reflected in the number of AFS cells available.

Secondly, AFS is designed with scalability in mind. If demand for AFS access increases then the number of AFS servers for a particular AFS cell can also be increased to handle this.

7.6 Demands On Users

A lot of time and effort has been spent developing protocols and interfaces that allow processes to communicate in these grid environments. There is a worry that in concentrating on the system/application level protocols, the end user has been left out of the loop. To expect the scientist to first become familiar with the grid environment before any science can be studied is a hassle. This could be reduced by making the grid an extension of the local computing environment. For example, one could introduce a standard way to access files locally and remotely, or introduce a standard way to run computation locally and remotely etc. Currently there are a lot of grids with a lot of different technologies and a lot of different interfaces.

The community has two long term approaches:

1. Make the protocols necessary to access the grid simple enough that any users can exploit any grid through their favourite interface. This would be the web

service approach currently at the cutting edge of eScience.

2. Make the interfaces universal such that each one is familiar already to the user community. The Web is fairly well understood, (grid portals would be an example e.g the NGS Portal [88]). File system access is familiar to most, indeed, access to AFS is now common enough that the current operating system of choice [1] contains both the client and the server.

7.7 Robust Software

The main server side component software used in this thesis are OpenAFS, Apache, Globus and `gssklogd`. There is a question of whether these software are secure and robust enough to be used in public and hostile networks. Certainly AFS and Apache have stood the test of time and have a huge community base behind them. Globus and gssklogd are fairly new and indeed early versions of both have seen problems. However, `gssklog` is based entirely on code derived from `klog` (OpenAFS) and Globus. With Globus a requirement for the HEP grids and AFS having been greatly tested and having much support from the community[2] there is little risk in using these technologies over what is already judged to be safe.

[1] Scientific Linux 3 [89]

[2] `gssklog` is also being tested in the OpenAFS community as it supports ticket granting based on kerberos credentials as well as GSI.

Chapter 8

Conclusion

AFS is a suitable file system for High Energy Physics Applications to run on within a grid. Long computational jobs with low levels of operational file I/O compared to compute time and local data I/O should have little observable impact. Delays of up to two minutes have been observed on real applications running on nodes at distances of 250 km from the AFS file store. In this case the speed-up observed when using a the higher specification machines at RAL far outweighed any time delay in all but the very short jobs.

This, however, still misses the main point that the transfer of analysis data to the point of submission is still largely an onerous task and far outweighs any marginal data transfer overheads observed in this thesis. AFS is certainly not a file system to distribute raw event data records; these either need to be located near to where the job runs or be small enough to be copied on demand and stored locally using some other technology. For such data transfers one might consider the Storage Resource Broker [21] or LCG SRM [90].

Current grid technologies lack a usability level that is required to make an analyst comfortable when using the grid. The time taken to become familiar with grid middleware is large. Things are made easier when the grid technologies are

abstracted away from the user, although this abstraction stifles the continual development of technology itself. The use of a global file system (such as AFS) that is accessed in a familiar way, similar to standard file system access, is one way of reducing the level of complexity that the user will experience when using the grid.

A benchmark was created to test the performance of file read, write and append operations. A command-line submission tool, `gsub`, which automatically obtains AFS credentials and sets up an execution environment suitable for HEP applications to run (specifically BaBar in this case) was written and presented. An on-line monitoring system was developed using a modified web server, GridSite (to which the Author contributed the GSI enabling code). Further command-line tools were created to keep track of jobs running on this grid and update credentials.

References

[1] Howard, J.H. An Overview of the Andrew File System in *Winter 1988 USENIX Conf. Proc., Dallas* February, 1988.
Online:
`http://reports-archive.adm.cs.cmu.edu/anon/itc/CMU-ITC-062.pdf`
[Accessed 20 Feb 2005]

[2] The Globus Toolkit®
Online: `http://www.globus.org`
[Accessed 14 Mar 2005]

[3] Grimshaw, A.S. Natrajan, A. Humphrey, M.A. Lewis, M.J. Nguyen-Tuong, A. Karpovich, J.F. Morgan, M.M. Ferrari, A.J. From Legion to Avaki: the persistance of vision. In Berman, F. Fox, J.C. Hey A.J.G. (eds.) *Grid Computing Making the Global Infrastructure a Reality*. Wiley, 2003, 265-298.

[4] Indiana University, Extreme Lab. Common Component Architecture Toolkit, Glossary
Online: `http://www.extreme.indiana.edu/ccat/glossary.html`
[Accessed 23 Feb 2005]

[5] NASA Information Power Grid, Glossary
Online: `http://www.ipg.nasa.gov/aboutipg/glossary.html`
[Accessed 23 Feb 2005]

[6] Foster, I. Tuecke, S. Introduction to the Grid Tutorial at *the first Global Grid Forum*, March 2001
Online: `http://www.ggf1.nl/presentations/Ian_Foster_1/ppframe.htm`
[Accessed 23 Feb 2005]

[7] Foster, I. Introduction to the Grid Part 1. Presentation at Global Grid Forum 1. 4-7 March 2000
Online: `http://www.globus.org/about/news/GriPhyN.html`
[Accessed 23 Feb 2005]

[8] Mc Keown, M. Jones, MAS. Viljoen, M. Installing Globus GT2 and Joining the L2G, Tutorial Slides *Presented at the National eScience Centre* April 2004.
Online: `http://www.sve.man.ac.uk/Training/Materials_Repository/mmk-GT2.ppt`
[Accessed 14 March 2005]

[9] Snelling, D. van den Berghe, S. Grid Technology and Roadmap from UNICORE to OGSA *Fujutsu 55-2*, March 2004, 152-168
Online: `http://magazine.fujitsu.com/vol55-2/paper08.pdf`
[Accessed 20 Feb 2005]

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

[11] Introduction to Grid Computing, (Tutorial).
Online: `http://www.globus.org/about/events/US_tutorial/slides/Intro-01-Grids3.pdf`
[Accessed 20 Feb 2005]

[12] Foster, I. Geisler, J. Nickless, B. Smith, W. Tuecke, S. Software Infrastructure for the I-WAY High-Performance Distributed Computing Experiment. In *Proceeeding of the Fifth IEEE Symposium on High Performance Distributed Computing.* IEEE Computer Society Press, 1996.
Online: `http://citeseer.ist.psu.edu/foster96software.html`
[Accessed 17 Feb 2005]

[13] Foster, I, Kesselman, C. Computational Grids. In Foster, I, Kesselman, C. (eds.) *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1998, 15-51.

[14] Foster, I, Kesselman, C. (eds.) *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1998.

[15] Stalling, W. *Cryptography and Network Security Principles and Practices*, Third Ed. Prentice Hall. pp 419. 2003

[16] Allcock, W. *et.al.* GFD.20-R-P GridFTP: Protocol Extensions to FTP for the Grid, January 2004.
Online: `http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf`
[Accessed 06 Feb 2005]

[17] UNICORE Plus Final Report - Uniform Interface to Computing Resources, Joint Project Report for the BMBF Project UNICORE Plus from January 2000 to December 2002

Online:
http://www.unicore.org/documents/UNICOREPlus-Final-Report.pdf
[Accessed 03 April 2005]

[18] GridEngine, Sun Microsystems Inc.
Online: http://gridengine.sunsource.net/
[Accessed 22 Jun 2005]

[19] Sun N1 Grid Engine 6, Sun Microsystems Inc.
Online: http://www.sun.com/software/gridware/index.xml
[Accessed 22 Jun 2005]

[20] Thain, D. Tannenbaum, T. & Livny, M. Distributed Computing in Practice: The Condor Experience *Concurrency and Computation: Practice and Experience* 2004.
Online: http://www.cs.wisc.edu/condor/doc/condor-practice.pdf
[Accessed 22 Jun 2005]

[21] Rajasekar, A. Wan, M. Moore, R. MySRB & SRB - Components of a Data Grid. *The 11th International Symposium on High Performance Distributed Computing (HPDC-11)* Edinburgh, Scotland, July 2002

[22] White Paper: The Entropia Approach to Distributed Computing
Online: http://www.entropia.com/
[Accessed 03 April 2005]

[23] Web Service, W3C Glossary
Online: http://www.w3.org/2003/glossary/keyword/All/?keywords=web%20service
[Accessed 22 Jun 2005]

[24] Cerami,E. Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. O'Rielly. Febuary 2002

[25] The OASIS Standards body web site.
Online: http://www.oasis-open.org/home/index.php
[Accessed 22 Jun 2005]

[26] Atkinson, M. Grid & Web services: Web Service Resource Framework WSRF Presentation at *Town Meeting April 04: Grids and Web Services* 23 April 2004
Online:
http://www.ogsadai.org.uk/docs/OtherDocs/GridsAndWebServices.pdf
[Accessed 24 Feb 2005]

[27] De Roure, D. Baker, M.A. Jennings, N.R. Shadbolt, N.R. The evolution of the Grid. In Berman, F. Fox, J.C. Hey A.J.G. (eds.) *Grid Computing Making the Global Infrastructure a Reality.* Wiley, 2003, 65-100.

[28] Eidelman, S. et al., *The Review of Particle Physics* Phys. Lett. B 593, 1. 2004. Online: `http://pdg.lbl.gov/`
[Accessed 22 Jun 2005]

[29] The LHC Computing Challenge *PPARC Website*
Online: `http://www.pparc.ac.uk/Rs/Pp/Sp/PPATSRannx.asp`
[Accessed 23 Jun 2005]

[30] Lueking, L. Loebel-Carpenter, L. Merritt, W. Moore, C. Pordes, R. Terekhov, I. Veseli, S. Vranicar, M. White, S. White, V. The D0 Experiment Data Grid - SAM. In Lee, C.A. (ed.) *Grid Gomputing - GRID 2001 Second International Workshop* Proceedings. Springer. November 2001, 177-184.

[31] Knobloch, J. Robertson, L. (eds.) LHC Computing Grid Technical Design Report version 1. LCG-TDR-001, CERN-LHCC-2005-024, 20 Jun 2005.
Online: `http://lcg.web.cern.ch/LCG/tdr/LCG_TDR_v1_02.pdf`
[Accessed 22 Jun 2005]

[32] Harvey, J. Computing Conference goes to the Swiss Alps *International Journal of High Energy Physics, CERN Courier*. Institute of Physics Publishing Ltd, Volume 45 Nr 1. 2005

[33] All SPEC CINT2000 *SPEC*
Online: `http://www.spec.org/cpu2000/results/cint2000.html`
[Accessed 17 March 2005]

[34] Pre-WS GRAM Documentation
Online:
`http://www-unix.globus.org/toolkit/docs/3.2/gram/prews/index.html`
[Accessed 18 March 2005]

[35] RSL Attributes
Online: `http://www-unix.globus.org/api/c-globus-3.2/globus_gram_documentation/html/`
[Accessed 18 March 2005]

[36] Load Sharing Facility, Platform Computing Inc.
Online: `http://www.platform.com/products/LSF/`
[Accessed 22 Jun 2005]

[37] Portable Batch System, Altair Grid Technologies
Online: `http://www.openpbs.org/`
[Accessed 22 Jun 2005]

[38] Network Queuing Environment, Silicon Graphics, Inc. and Cray Research, Inc.
Online: `http://docs.cray.com/books/2148_3.3/html-2148_3.3/`
[Accessed 22 Jun 2005]

[39] The Globus Resource Specification Language RSL v1.0. Webpage
Online: `http://www.globus.org/gram/rsl_spec1.html`
[Accessed 18 March 2005]

[40] Bester,J. Foster, I. Kesselman, C. Tedesco, J. Tuecke S. A Data Movement and Access Service for Wide Area Computing Systems. *Sixth Workshop on I/O in Parallel and Distributed Systems*, May 5, 1999.

[41] Stalling, W. *Cryptography and Network Security Principles and Practices*, Third Ed. Prentice Hall.pp 402. 2003

[42] McDaniel, E. AFS File Sharing. *Guide to Eos and Unity Computing 2004-05 Edition for UNIX, Windows, and Linux.*115-130
Online: `http://www.eos.ncsu.edu/guide/pdf/afs.pdf`
[Accessed 25 Feb 2005]

[43] McNab, A. Grid-based access control and user management for Unix environments, Filesystems, Web Sites and Virtual Organisations. Presented at *CHEP 2003, La Jolla, California* March 2003.
Online: `http://www.gridpp.ac.uk/papers/chep03_TUBT008.PDF`
[Accessed 18 March 2005]

[44] Kornievskaia, O. Honeyman, P. Doster, B. Coffman, K, Kerberized Credential Translation: A Solution to Web Access Control, *USENIX Security Symposium, Washington, D.C.* August 2001.
Online:
`http://www.citi.umich.edu/techreports/reports/citi-tr-01-5.pdf`
[Accessed 23 Feb 2005]

[45] Novotny, J. Tuecke, S. and Welch, V. An Online Credential Repository for the Grid: MyProxy. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[46] RFC 2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Housley, R. Ford, W. Polk, W. Solo, D. January 1999

[47] Welch, V. Foster, I. Kesselman, C. Mulmo, O. Pearlman, L. Tuecke, S. Gawor, J. Meder, S. Siebenlist, F. X.509 Proxy Certificates for Dynamic Delegation. *3rd Annual PKI R&D Workshop*, 2004.
Online: `http://www.globus.org/Security/papers/`
`pki04-welch-proxy-cert-final.pdf`
[Accessed 23 Feb 2005]

[48] RFC3820 Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile; Tuecke, S. Welch, V. Engert, D. Pearlman, L. Thompson, M. Jun 2004

[49] GSI enabled `klog` Source Code
Online: `ftp://achilles.ctd.anl.gov/pub/DEE/old/gsiklog-0.9.tar`
[Accessed 04 Apr 2005]

[50] GSI enabled `klog` Source Code
Online: `ftp://achilles.ctd.anl.gov/pub/DEE/gssklog-0.11.tar`
[Accessed 04 Apr 2005]

[51] Freier, A.O. Karlton, P. Kocher, P.C. The SSL Protocol Version 3.0 *Transport Layer Security Working Group* INTERNET-DRAFT, 18 November 1996
Online: `http://wp.netscape.com/eng/ssl3/`
[Accessed 23 Feb 2005]

[52] Barlow, R.J. Forti, A. McNab, A. Salih, S. Smith, D. Ayde, T. BaBar Web job submission with Globus authentication and AFS access. Presented at *Computing in High Energy and Nuclear Physics* 24-28 March 2003
Online: `http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOAT011.PDF`
[Accessed 06 Mar 2005]

[53] GridSite Website
Online: `http://www.gridsite.org/`
[Accessed 03 Mar 2005]

[54] Engelschall, R.S. Security Solutions with SSL. Presented at *ApacheCon*, Santa Clara, April 4th 2001.
Online: `http://www.modssl.org/docs/apachecon2001/`
[Accessed 06 Mar 2005]

[55] RFC1945 Hypertext Transfer Protocol – HTTP/1.0. Berners-Lee, T. Fielding, R. Frystyk, H. May 1996

[56] RFC2616 Hypertext Transfer Protocol – HTTP/1.1. Fielding, R. Gettys, J. Mogul, J. Frystyk, H. Masinter, L. Leach, P. Berners-Lee, T. June 1999

[57] Cole, K.J. Hewitt, W.T. Jones, M.A.S. Pickles, S.M. Riding, M. Roy, K, Russell, C. Sensier, M. Seamless Access to Multiple Datasets (SAMD) - An ESRC e-Science Demonstrator. Presented at the *ESRC Research Methods Workshop Combining data: using advanced technology to enhance social science resources*, Manchester, 18 December, 2002.
Online: `http://www.ccsr.ac.uk/methods/archive/combiningdata/samd_dec18.pdf`
[Accessed 06 Mar 2005]

[58] SAMD Website
Online: `http://www.sve.man.ac.uk/Research/AtoZ/SAMD/`
[Accessed 24 Jun 2005]

[59] Curl, A Client for URLs
Online: `http://curl.haxx.se/`
[Accessed 20 Mar 2005]

[60] Zeldovich, N. Rx protocol specification draft.
Online: `http://web.mit.edu/kolya/afs/rx/rx-spec`
[Accessed 20 Mar 2005]

[61] RFC793 Transmission Control Protocol; Information Sciences Institute. Sep 1981

[62] RFC768 User Datagram Protocol; Postel, J. Aug 1980

[63] Bachmann, D. Honeyman, P. Huston, L.B. The Rx Hex *1st International Workshop on Services in Distributed and Networked Environments* IEEE Computer Society Press, 1994, 66-74.
Online: `http://citeseer.ist.psu.edu/bachmann94rx.html`
[Accessed 21 Mar 2005]

[64] Gupta, R. Ansari, S. Les Cottrell, R. Hughes-Jones, R. Characterization and Evaluation of TCP and UDP-based Transport on Real Networks. Presented at *Third International Workshop on Protocols for Fast Long-Distance Networks*. February 2005
Online: `http:`
`//www.ens-lyon.fr/LIP/RESO/pfldnet2005/Papers/1_2_Cottrell_R.pdf`
[Accessed 20 March 2005]

[65] Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., and West, M.J. Scale and Performance in a Distributed File System. ACM Transactions on Computer Systems, 6(1), February 1988.
Online: `http://www-2.cs.cmu.edu/afs/cs/project/coda-www/`
`ResearchWebPages/docdir/s11.pdf`
[Accessed 13 Mar 2005]

[66] The Andrew Benchmark, Original version by M. Satyanarayanan,School of Computer Science, Carnegie-Mellon University,1985. Modified 2001
Online:
`http://www.citi.umich.edu/projects/nfs-perf/results/cmarion/`
[Accessed 13 Mar 2005]

[67] Press, B. Latency and Jitter on Your LAN
Online: `http://www.informit.com/articles/article.asp?p=20020&rl=1`
[Accessed 23 Jun 2005]

[68] Arla AFS client, Project Web Page
Online: `http://www.stacken.kth.se/projekt/arla/`
[Accessed 03 Mar 2005]

[69] Gnu BASH (website)
Online: `http://www.gnu.org/software/bash/bash.html`
[Accessed 23 Jun 2005]

[70] The Perl Foundation website
Online: `http://www.perlfoundation.org/`
[Accessed 23 Jun 2005]

[71] Python (website)
Online: `http://www.python.org/`
[Accessed 23 Jun 2005]

[72] Andreozzi, S. The Glue Schema (Website)
Online: `http://www.cnaf.infn.it/~sergio/datatag/glue/`
[Accessed 22 Mar 2005]

[73] HEPiX Shells Scripts - Files Web Page
Online: `http://wwwhepix.web.cern.ch/wwwhepix/wg/scripts/www/shells/files.html`
[Accessed 22 Mar 2005]

[74] HEPiX FAQ
Online: `http://wwwhepix.web.cern.ch/wwwhepix/FAQ`
[Accessed 22 Mar 2005]

[75] Gaudi/Athena and Grid Alliance website.
Online: `http://ganga.web.cern.ch/ganga/`
[Accessed 22 Mar 2005]

[76] Jones, M.A.S. Barlow, R.J. Forti, A. AliBaBa: Running BaBar jobs on the grid using gsub (AFS Access to Local Installations of BaBar) Cox, S.J. (ed.) *Proceedings of the UK e-Science All Hands Meeting* Nottingham, UK Aug/Sep 2004
Online: `http://www.allhands.org.uk/2004/proceedings/papers/272.pdf`
[Accessed 4 Apr 2005]

[77] Jones, M.A.S. Barlow, R.J. Forti, A. McNab, A. AliBaBa: A Heterogeneous Grid-based Job Submission system used by the BaBar Experiment. At CHEP04, Interlaken Switzerland, Sep/Oct 2004
Online: `http://indico.cern.ch/getFile.py/access?contribId=337&sessionId=23&resId=1&materialId=paper&confId=0`
[Accessed 4 Apr 2005]

[78] The BaBar homepage
Online: `http://www.slac.stanford.edu/BFROOT`
[Accessed 4 Apr 2004]

[79] Boutigny et. Al. on behalf of the BaBar Computing Group, Use of the European Data Grid software in the framework of the BaBar distributed computing model. *Conference Proceedings CHEP03* La Jolla. 2003

[80] Dorigo, A. BBRORA and skimData Webpage. June 2001
Online: `http://www.slac.stanford.edu/BFROOT/www/Computing/Offline/DataDist/KDDIBT.html`
[Accessed 04 Apr 2005]

[81] skimData Web Interface
Online: `http://bfhome.hep.man.ac.uk/skimData.pl`
[Accessed 15 Oct 2004]

[82] Elmer, P. CM2 - An Introduction.
Online: `http://www.slac.stanford.edu/BFROOT/www/Computing/Documentation/CM2/intro/`
[Accessed 18 Jun 2005]

[83] BbkDatasetTcl Online Documentation
Online: `http://www.slac.stanford.edu/BFROOT/www/Computing/Distributed/Bookkeeping/Documentation/BbkDatasetTcl.html`
[Accessed 22 Jun 2005]

[84] European DataGrid. Web page
Online: `http://eu-datagrid.web.cern.ch/eu-datagrid/`
[Accessed 04 Apr 2005]

[85] Large Hadron Collider Computing Grid. Web page
Online: `http://lcg.web.cern.ch/LCG`
[Accessed 04 Apr 2005]

[86] OpenAFS UserGuide
Online: `http://www.openafs.org/pages/doc/UserGuide/auusg010.htm`
[Accessed 23 Mar 2005]

[87] BaBarGrid Hypernews Private Mailing List Thread, August 2003
Online: `http://babar-hn.slac.stanford.edu:5090/HyperNews/get/BaBarGrid/239/1/1/1.html`
[Accessed 22 Mar 2005]

[88] The National Grid Service Portal
Online: `https://portal.ngs.ac.uk/`
[Accessed 22 Jun 2005]

[89] Sieh, C. Sliding into Free Enterprise Linux. Presentation at *Spring 2004 HEPiX meeting* the eScience Institute, Edinburgh, 24-28 May, 2004
Online: `http://hepwww.rl.ac.uk/hepix/nesc/sieh1.pdf`

[Accessed 22 Mar 2005]

Streaming Media
Online: `http://www.net.rl.ac.uk/ramfiles/hepix2004-9b.ram`
[Accessed 22 Mar 2005]

[90] Storage Resource Management Working Group Webpage
Online: `http://sdm.lbl.gov/srm-wg/`
[Accessed 24 Jun 2005]

[91] Foster, I. Kesselman C. The Globus Toolkit. In Foster, I. Kesselman C. (eds) *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1998, 259,278.

[92] The Ephemeral Port Range
Online: `http://www.ncftpd.com/ncftpd/doc/misc/ephemeral_ports.html`
[Accessed 22 Jun 2005]

[93] The AFS-NFS Comparison originally from `ftp://ftp.transarc.com/pub/afsps/doc/afs-nfs.comparison`
Online: `http://grand.central.org/twiki/bin/view/AFSLore/GeneralFAQ`
[Accessed 02 Mar 2005]

[94] Center for Parallel Computers, Royal Institute of Technology, Swedish, Kerberos Web Page
Online: `http://www.pdc.kth.se/kth-krb/`
[Accessed 03 Mar 2005]

[95] KTH-Kerberos Source Code
Online: `ftp://ftp.pdc.kth.se/pub/krb/src/krb4-1.2.2.tar.gz`
[Accessed 03 Mar 2005]

[96] Arla AFS client Source Code
Online: `ftp://ftp.stacken.kth.se/pub/arla/arla-0.36.2.tar.gz`
[Accessed 03 Mar 2005]

Glossary and Acronyms

ACL	Access Control List
AFS	A distributed file system formerly called the Andrew File System
afsfsperf	A test AFS client from ARLA
afslog	A command to obtain an AFS token from a TGT
AJO	Abstract Java Object; a component of UNICORE
Arla	A free AFS client. Arla is swedish for "early" and also a dairy supplier; "filmjölk" is swedish for both sour milk and file!
BaBar	A Particle Physics experiment at SLAC
Beowulf Cluster	A cluster of PCs usually with an ethernet based interconnect
BetaMiniApp	The Basic BaBar analysis code
BFROOT	An environment variable pointing to the directory containing the BaBar releases
CA	Certificate Authority; a source of authority allowing on-line entities to establish their identity issuing X.509 certificates
CellServDB	File containing IP addresses of all configured AFS servers and their corresponding cell on a Client Machine
certificate request	A file containing a public key to be signed by a CA.

CGI	Common Gateway Interface; allows programs to be executed on a web server.
Compute Element (CE)	EDG/LCG terminology for a gatekeeper or server running a Globus jobmanager.
Condor	A workload management system from the University of Wisconsin-Madison
CRL	Certificate Revocation List
Curl	A Client for accessing URLs; a command to access e.g. websites using either http and https
Dæmon	A dormant program that waits for a condition to occur before waking to perform a task, returning to its dormant state afterwards. Now realised as the Acronym: "Disk And Execution MONitor"
DES	Data Encryption Standard
DESY	Deutsches Elektronen-Synchrotron, A particle Accelerator facility in Hamburg Germany.
Dump	Utility to read a TCL script and convert it and all dependencies into one large TCL file.
EDG	The European DataGrid
ephemeral port	A port opened (briefly) by a client to listen for connections from a server. Connections are usually initiated by a command from the client to the server. Any unreserved port can be used as an ephemeral port, usually this means the port number is between 1025 and 65535 although operating systems may restrict this further.[92]
Fedora Core	A non-enterprise version of Linux
FNAL	Fermi National Accelerator Laboratory, Near Chicago, USA
FTP	File Transfer Protocol
GACL	Grid Access Control List

Ganga	Gaudi/Athena and Grid Alliance. User interface to the grid
GASS	Global Access to Secondary Storage
gdel	A command to delete grid jobs
GIIS	Grid Information Index Service
Globus Allience	A project based initially at the University of Chicago and Argonne National Lab to research and develop grid middleware
Globus Toolkit®	The middleware produced by the Globus Alliance
gnew	A command to push an updated proxy to all running jobs (in order to obtain a new AFS credential)
GRAM	Grid Resource Allocation Manager
Grid	Defined in Chapter 1.1
GridEngine	A workload management system based on Codine, also available as Sun Grid Engine
GridFTP	Grid enabled FTP allowing GSI authenticated control and data channels and parallel data stream
GridPP	The UK HEP Grid Collaboration
GSI	Grid Security Infrastructure
gsiklog	Client tool for obtaining an AFS token using a GSI credential
GSSAPI	Generic Security Services Application Programme Interface
gssklog	Client tool for obtaining an AFS token using either a Kerberos ticket or GSI credential using the GSSAPI
gstat	A command to obtain grid job statuses
gsub	Grid submission programme allowing access to AFS access

GUI	Graphical User Interface
HEP	High Energy Physics
HTCP	HyperText Copy
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol, HTTP over SSL
hypernews	Online Message and news server
I/O	Input/Output
IP	Internet Protocol
I-POP	I-WAY Point of Presence: An agent server in the I-WAY project
I-WAY	Information Wide Area Year, A metacomputing project spread across 17 institutes
Job	A computer code sent to a computer to be run.
Kerberos	A Network security infrastructure based on shared keys
kinit	A command to obtain a TGT
klog	A command used to obtain an AFS token from an AFS KA Server
LCG	The LHC Computing Grid
LDAP	Lightweight Directory Access Protocol
LHC	Large Hadron Collider
libc	C runtime library
LSF	Load Sharing Facility, a batch system from Platform

MDS Officially: The Globus Toolkit®Monitoring and Discovery System [2]. Also referred to as the Metacomputing and Directory Service [91], Meta Directory Service, Monitoring and Directory Service, MetaData Service.

Metacomputing The use of many computers distributed geographically as one computer

Metadata Data about data

mk-gridmapfile Command to construct a Globus gridmapfile for pool accounts from VO memberships

NASA IPG The NASA Information Power Grid

NIC Network Interface Card

NFS The Network File System

NGS National Grid Service, a production grid in the UK currently comprising of three JISC funded and one RAL funded Linux clusters plus two national supercomputers

NNPFS Kernel Module for the Arla AFS client previously known as XFS

NQE Network Queuing Environment, a batch system from Cray Research

OGSA Open Grid Services Architecture

OGSI Open Grid Services Infrastructure

OpenAFS A free implementation of AFS based on Transarc AFS

PAM Plugable Authentication Module

PBS Portable Batch System, from Altair Engineering, also available in an open source version OpenPBS

Petabyte (PB) (There are two different definitions of Petabyte: technically 2^{50} bytes ($\approx 1.13 \times 10^{15}$ bytes), popularly 1×10^{15} bytes.

ping	Packet INternet Groper, a Unix command to check the IP connection.
PKI	Public Key Infrastructure
PTS	Protection Server. A Component of AFS server.
qsub	PBS queue submission programme
RSL	Resource Specification Language
RTT	Round Trip Time
SAM	Sequential Access through Meta-data
Sandbox	HEP: Filespace used and required for running applications. Computer Science: Secure execution environment to protect the operating system.
SAN	Storage Area Network
skimdata	Command to obtain datasets in the first BaBar Computing Model.
SLAC	Stanford Linear Accelerator Center, near San Francisco USA
SOA	Source of Authority, eg a Certificate Authority, also Service Oriented Architecture
SRB	Storage Resource Broker
SRM	Storage Resource Manager
SSL	Secure Socket Layer
SUSY	Super Symmetry - One theory beyond the Standard Model
Tcl	Tool Command Language, a simple scription language
TCP	Transmission Control Protocol

TeraFLOPs	1×10^{12} floating point operations per second
TGS	Ticket Granting Server, Kerberos
TGT	Ticket Granting Ticket, Kerberos
Transarc AFS	An implementation of AFS
TSI	Target System Interface, UNICORE Scripted access to batch systems on target machines
UDP	User Datagram Protocol
UNICORE	Uniform access to Computer Resources
U-Site	UNICORE Site. Accessable through one Gateway
VDT	Virtual Data Toolkit
VO	Virtual Organisation
WN	Worker nodes, also known as back-end-nodes, are target machines that the CE will distribute jobs to.
WSI	Web Service Infrastructure
WSRF	Web Service Resource Framework

Appendix A

Comparison Of AFS To NFS

Table A is a copy of the "afs-nfs.comparison" formerly available from [93]. The original is no longer available in its original form and so has been included here.

	AFS	NFS
File Access	Common name space from all workstations	Different file names from different workstations
File Location Tracking	Automatic tracking by file system processes and databases	Mountpoints to files set by administrators and users
Performance	Client caching to reduce network load; callbacks to maintain cache consistency	No local disk caching; limited cache consistency
Andrew Benchmark (5 phases, 8 clients)[1]	Average time of 210 seconds/client	Average time of 280 seconds/client
Scaling capabilities	Maintains performance in small and very large installations	Best in small to mid-size installations
	Excellent performance on wide-area configuration	Best in local-area configurations
Security	Kerberos mutual authentication	Security based on unencrypted user ID's
	Access control lists on directories for user and group access	No access control lists
Availability	Replicates read-mostly data and AFS system information	No replication
Backup Operation	No system downtime with specially developed AFS Backup System	Standard UNIX backup system
Reconfiguration	By volumes (groups of files)	Per-file movement
	No user impact; files remain accessible during moves, and file names do not change	Users lose access to files and filenames change (mountpoints need to be reset)
System Management	Most tasks performed from any workstation	Frequently involves telnet to other workstations
Autonomous Architecture	Autonomous administrative units called cells, in addition to file servers and clients	File servers and clients
	No trust required between cells	No security distinctions between sites

Table A.1: Comparison of AFS to NFS (verbatim [93])

[1]The original benchmark

Appendix B

Arla Installation

Unlike OpenAFS or Transarc AFS, Arla is an independent AFS implementation written bottom up by the open source community.

This is a walk through of the installation of Arla on my desktop PC and laptop. They both run Fedora Core 1 installed with the 2.4.22-1.2197.nptl stock kernel.

Prerequisites

- A Kerberos IV implementation. Arla recommend kth-KRB [94]. I used version 1.2.2 [95].
- Arla [68]. I used 0.36.2 [96].

Installing The Kerberos IV Implementation

This will install a Kerberos IV into the directory `/usr/athena`

```
$ wget ftp://ftp.pdc.kth.se/pub/krb/src/krb4-1.2.2.tar.gz
$ tar -xzvf krb4-1.2.2.tar.gz
$ cd krb4-1.2.2
$ ./configure
```

Fedora core ships with openssl 0.9.7 installed, krb4-1.2.2 is designed around version 0.9.6. The definitions of various structs is different in the newer openssl implementation and so kth-krb doesn't compile. Add the following lines to the end of the file `include/config.h`:

```
#define HAVE_H_ERRNO 1
#define HAVE_H_ERRNO_DECLARATION 1
#define OPENSSL_DES_LIBDES_COMPATIBILITY 1
#define des_ks_struct _ossl_old_des_ks_struct
```

You'll also need to change one of the function calls in `lib/roken/ndbm_wrap.c`. Line 168 should read:

```
if(db->open(db, fn, NULL, NULL, DB_BTREE, myflags, mode) != 0) {
```

in place of

```
if(db->open(db, fn, NULL, DB_BTREE, myflags, mode) != 0) {
```

Make and install it:

```
$ make
$ su -c "make install"
$ cd ..
```

Configuring Kerberos IV

Two files need information about your Kerberos KA Server: `/etc/krb.realms` and `/etc/krb.conf`. These files may already be installed if you have krbafs (shipped with Fedora). You should add the following to the start of your `/etc/krb.realms` file:

```
mcc.ac.gb       MCC.AC.GB
.mcc.ac.gb      MCC.AC.GB
```

and you should add the following to the top of your `/etc/krb.conf` file (actually only the first line need go at the top, that represents the default realm):

```
MCC.AC.GB
MCC.AC.GB       scree.mcc.ac.uk
MCC.AC.GB       crag.mc.man.ac.uk
MCC.AC.GB       scarp.mc.man.ac.uk
```

This will allow you, when the time comes, to obtain a Ticket Granting ticket using athena's `kinit`, (klist to see the TGT) and an `afslog` to get an AFS token

Installing The Arla

This will make Arla and install it in `/usr/arla`

```
$ wget ftp://ftp.stacken.kth.se/pub/arla/arla-0.36.2.tar.gz
$ tar -xzvf arla-0.36.2.tar.gz
$ cd arla-0.36.2
$ ./configure --with-krb4=/usr/athena
$ make
```

make fails in the directory nnpfs/linux because we're using `gcc3.3.x`. Edit nnpfs/linux/Makefile and add an extra CFLAG: `-fno-strict-aliasing` and continue...

```
$ make
$ su -c "make install"
```

Configuring Arla

As per OpenAFS there is a ThisCell and CellServDB in /usr/arla/etc. You need to add your CellServer and ThisCell as appropriate. The CellServDB shipped with Arla contains very old servers for mcc.ac.gb, change the relevant lines to:

```
>mcc.ac.gb              #University of Manchester
130.88.203.64                   #scarp.mc.man.ac.uk
130.88.203.13                   #scree.mcc.ac.uk
130.88.203.10                   #crag.mc.man.ac.uk
```

ThisCell should contain the following single line:

```
mcc.ac.gb
```

Running Arla

Add the library directories to the ld.so.conf file and configure

```
$ su -
# echo "/usr/athena/lib" >> /etc/ld.so.conf
# echo "/usr/arla/lib" >> /etc/ld.so.conf
# ldconfig
# exit
```

Start it up and use it:

```
$ su -c "/usr/arla/sbin/startarla"
$ /usr/athena/bin/kinit username@MCC.AC.GB
$ /usr/athena/bin/klist   #to list the current tickets and tokens held
$ /usr/athena/bin/afslog
$ /usr/arla/bin/tokens    #to list the tokens
$ /bin/ls /afs/mcc.ac.gb  #to list the contents of /afs/mcc.ac.gb
```

Logging Onto Other realms

Here's an example of how to get tokens for the hep.man.ac.uk realm (if you have an account there, same applies for other realms. Stop arla (if no one's using it: check with `fuser /afs`)

```
$ su -
# umount /afs
# killall arlad
# rmmod nnpfs
# exit
```

Add the realm info to the various files /usr/arla/etc/CellServDB

```
>hep.man.ac.uk          \#Manchester HEP
194.36.2.3                      \#afs1.hep.man.ac.uk
194.36.2.4                      \#afs2.hep.man.ac.uk
194.36.2.6                      \#afs4.hep.man.ac.uk
```

Add lines to `/etc/krb.realms` and `/etc/krb.conf`

```
$ su -
# echo "hep.man.ac.uk   HEP.MAN.AC.UK" >> /etc/krb.realm
# echo "HEP.MAN.AC.UK   afs1.hep.man.ac.uk" >> /etc/krb.conf
# echo "HEP.MAN.AC.UK   afs2.hep.man.ac.uk" >> /etc/krb.conf
# echo "HEP.MAN.AC.UK   afs4.hep.man.ac.uk" >> /etc/krb.conf
# exit
```

Restart arla. and then get your new tokens:

```
$ /usr/athena/bin/kinit masj@HEP.MAN.AC.UK
$ /usr/athena/bin/afslog -cell hep.man.ac.uk
```

Appendix C

Write Read Append Benchmark

This chapter contains the code used to create a simple benchmark designed to test file write read and append speeds. Given that the Grid offers a diverse array of systems this benchmark needs to be executable at any site with a suitable AFS installation.

The benchmark is a number of compiled C programmes executed from a bourne shell script.

Suitable AFS credentials are required.

Wrapper Script

This wrapper script takes care of flushing the AFS volume before each benchmark. It controls how many files will be created in any one session, how larger they should be allowed to get and in which AFS volume to create them.

```
#!/bin/sh

#Set this to the spave vailable in the smallest volume
LIM=9000000

#Set this to control the number of files written
NUM=10

size=1
while [ `expr $size \* 2 \* $NUM` -lt $LIM ]
do
  for VOL in /afs/hep.man.ac.uk/u/masj \
             /afs/mcc.ac.gb/users/cgu/zzcgumj/Vol2 \
             /afs/rl.ac.uk/user/j/jonesmas \
             /afs/desy.de/user/j/jonesmas \
             /afs/slac.stanford.edu/u/br/jonesmas
  do

    DIR=$VOL/AFSTESTS-WRA
    rm -rf $DIR
    mkdir -p $DIR

    fs flushvolume $VOL
    a=`./gettime`
    ./wrawrite $NUM $size $DIR >/dev/null
    b=`./gettime`
    IN=`echo $a`
    OUT=`echo $b`
    TIME=`dc -e "$OUT $IN - p"`
    echo $VOL write $size $TIME

    fs flushvolume $VOL
    a=`./gettime`
    ./wraread $DIR >/dev/null
    b=`./gettime`
    IN=`echo $a`
    OUT=`echo $b`
    TIME=`dc -e "$OUT $IN - p"`
    echo $VOL read $size $TIME

    fs flushvolume $VOL
    a=`./gettime`
    ./wraappend $size $DIR >/dev/null
    b=`./gettime`
    IN=`echo $a`
    OUT=`echo $b`
    TIME=`dc -e "$OUT $IN - p"`
    echo $VOL append $size $TIME
  done
  size=`expr $size \* 2`
done
```

`wrawrite`

This benchmark takes three arguments: the number of files to create, the amount of data to place in each file and the directory of files to which this data will be added.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>

int main(int argc, char *argv[])
{
  int a,b,c,maxfiles,filesize;
  char file[128];
  FILE *fptr;
  unsigned char i='0';
  char path[128];
  char files[40];

  maxfiles=1024;
  if (argc != 4 ) exit(1);

  maxfiles=atoi(argv[1]);
  filesize=atoi(argv[2]);
  strcpy(path,argv[3]);
  printf("Testing %i files of size %i create and write to %s.\n", maxfiles, filesize, path);

  if (mkdir(path, 00777) == 0 ) { printf ("made dir %s.\n",path);}

  for (a=0;a<maxfiles;a++)
  {
    printf ("making file %04i\n",a);
    sprintf(file,"%s/%04i",path,a);
    if ( (fptr = fopen(file,"w")) == NULL )
    {
      exit(1);
    }
    else
    {
      for (b=0;b<filesize;b++)
      {
        fputc(i,fptr);
      }
      fclose(fptr);
    }
  }
return 0;
}
```

`wraread`

This benchmark takes one argument: the directory of files to read. It opens and reads the directory before reading the files, this operation will be included in the benchmark.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>

int main(int argc, char *argv[])
{
  int a,b,c,maxfiles,filesize;
  char file[140];
  FILE *fptr;
  DIR *DirectoryPointer;
  struct dirent *dp;
  unsigned char i='0';
  char path[128];
  char files[40];

  if (argc != 2 ) exit(1);

  strcpy(path,argv[1]);
  printf("Testing read of all files in %s.\n",  path);

  DirectoryPointer = opendir(path);
  for (dp = readdir(DirectoryPointer); dp != NULL; dp = readdir(DirectoryPointer))
  {
    if (dp->d_name[0] != '.')
    {
      strcpy(file,path);
      strcat(file,"/");
      strcat(file,dp->d_name);
      printf ("opening and reading %s ",file);
      if ( (fptr = fopen(file,"r")) == NULL )
      {
        printf("FAIL\n");
      }
      else
      {
        filesize=0;
        while (fgetc(fptr) != EOF) filesize++;
        printf("%i bytes read\n",filesize);
        fclose(fptr);
      }
    }
  }
  closedir(DirectoryPointer);
return 0;
}
```

`wraappend`

This benchmark takes two arguments: the amount of data to increase each file by and the directory of files to which this data will be added. It opens and reads the directory before writing the files, this operation will be included in the benchmark.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>

int main(int argc, char *argv[])
{
  int a,b,c,maxfiles,filesize;
  char file[140];
  FILE *fptr;
  DIR *DirectoryPointer;
  struct dirent *dp;
  unsigned char i='0';
  char path[128];
  char files[40];

  if (argc != 3 ) exit(1);

  filesize=atoi(argv[1]);
  strcpy(path,argv[2]);
  printf("Testing append %i bytes to all files in %s.\n", filesize, path);

  DirectoryPointer = opendir(path);
  for (dp = readdir(DirectoryPointer); dp != NULL; dp = readdir(DirectoryPointer))
  {
    if (dp->d_name[0] != '.')
    {
      strcpy(file,path);
      strcat(file,"/");
      strcat(file,dp->d_name);
      printf ("opening and appending %s ",file);
      if ( (fptr = fopen(file,"a")) == NULL )
      {
        printf("FAIL\n");
        exit(1);
      }
      else
      {
        for (b=0;b<filesize;b++)
        {
                fputc(i,fptr);
        }
        printf("%i bytes written\n",filesize);
        fclose(fptr);
      }
    }
  }
  closedir(DirectoryPointer);
return 0;
}
```

gettime

`gettime` is the same programme as used in the modified Andrew Benchmark. This code asks the system for the time stamp to the nearest μs.

```
#include <sys/time.h>
#include <unistd.h>
#include <stdio.h>

struct timeval tp;

main()
{
  gettimeofday(&tp, (struct timezone *) 0);
  printf("%ld.%06ld\n", tp.tv_sec, tp.tv_usec);
  exit (0);
}
```

Appendix D

`afsfsperf` Encrypted And Plain Tests

This appendix shows the results from the `afsfsperf` benchmark for the encrypted data and the unencrypted, unauthenticated data transactions not discussed in chapter 3. Figures D.1-D.5 show the `afsfsperf` tests run with encryption. Figures D.6-D.10 show the `afsfsperf` tests run without encryption and without authentication.

Figure D.2 shows that for read operations hep.man.ac.uk is as slow as rl.ac.uk. Encrypting the data has a large affect on the data transfer and seems to affect the data transfer in strange ways. This could be caused by the efficiency of the encryption/decryption algorithms used and the differing performance of the client and server. Without further detailed study it is hard to say with any certainty what is happening in this type of data transfer.

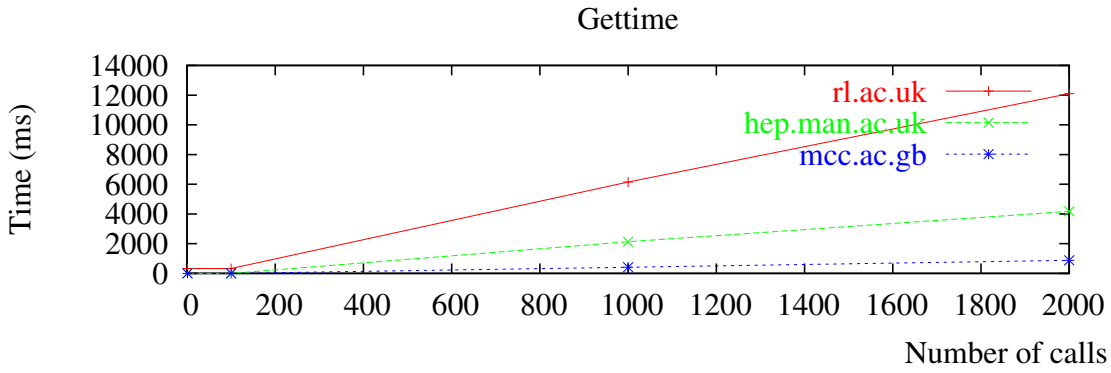


Figure D.1: `afsfsperf` encrypted time requests

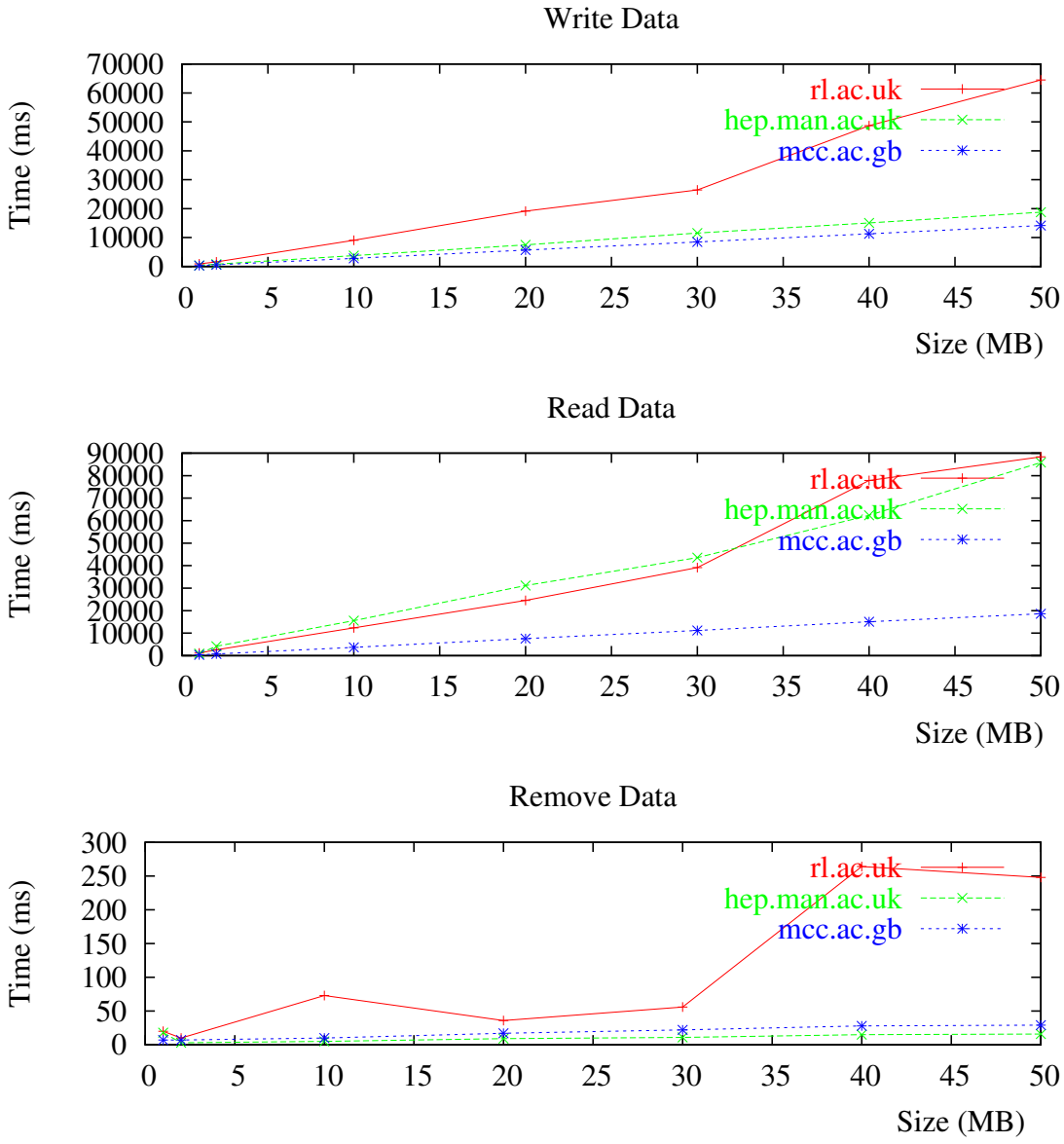


Figure D.2: `afsfsperf` encrypted data measurements

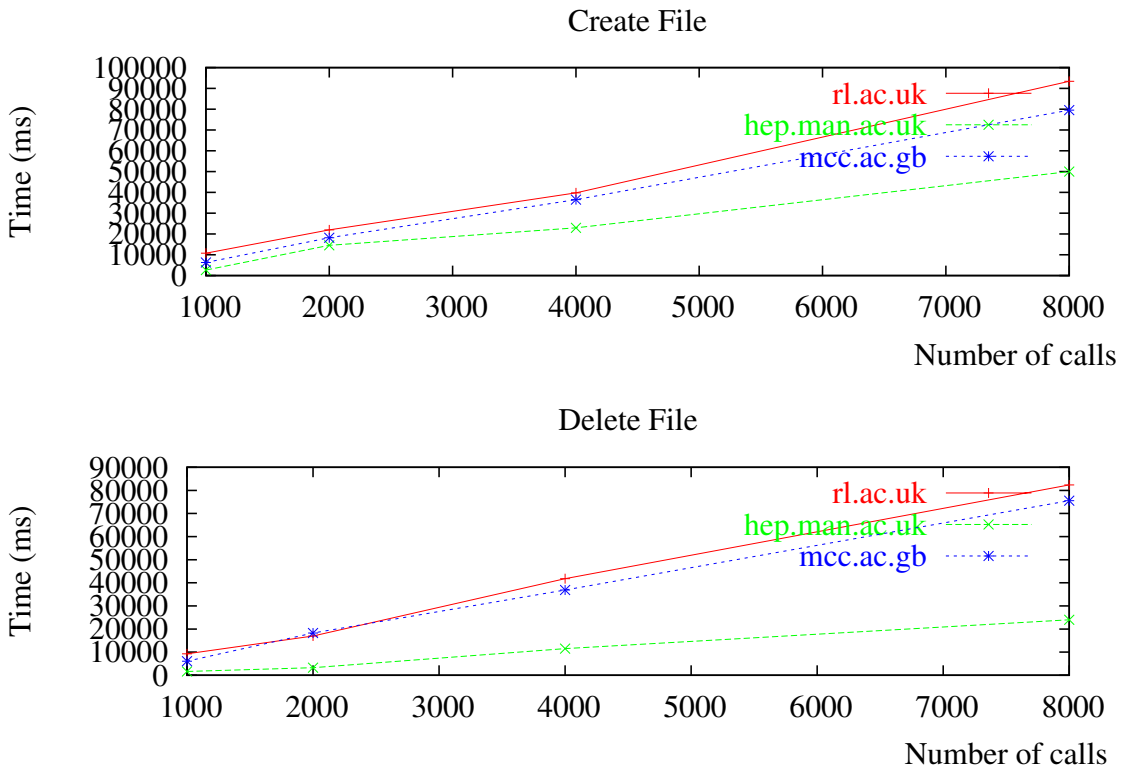


Figure D.3: `afsfsperf` encrypted file operations

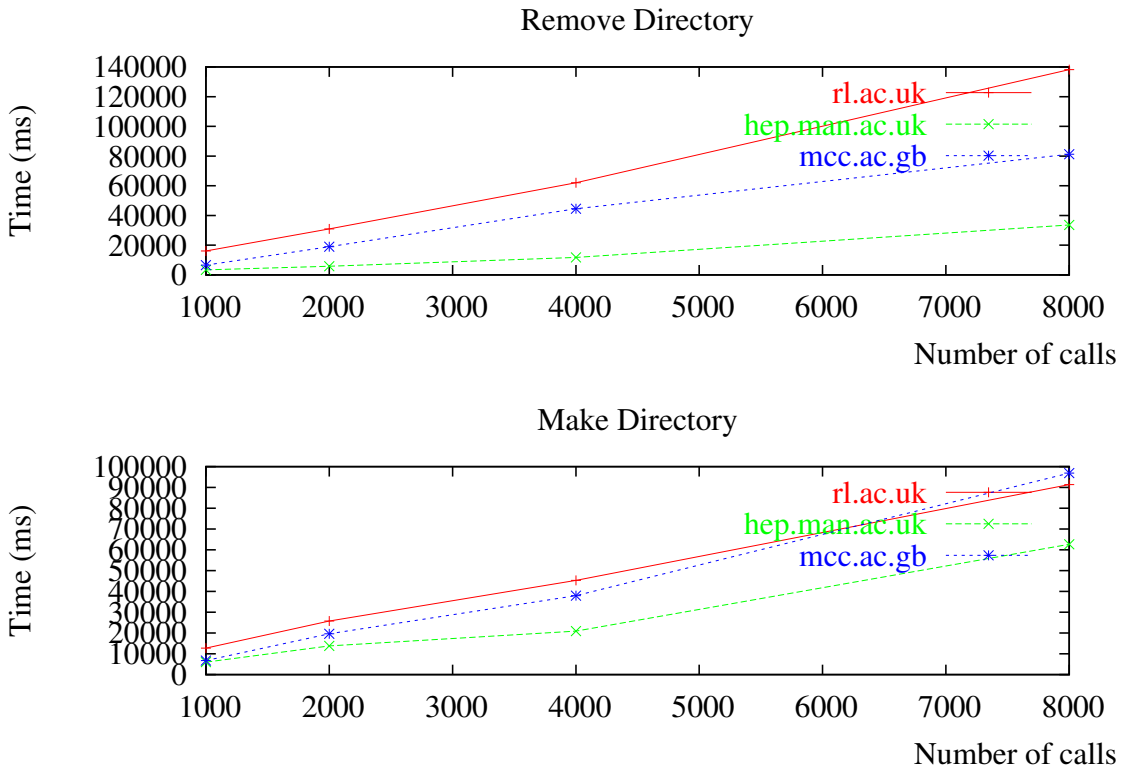


Figure D.4: `afsfsperf` encrypted directory operations

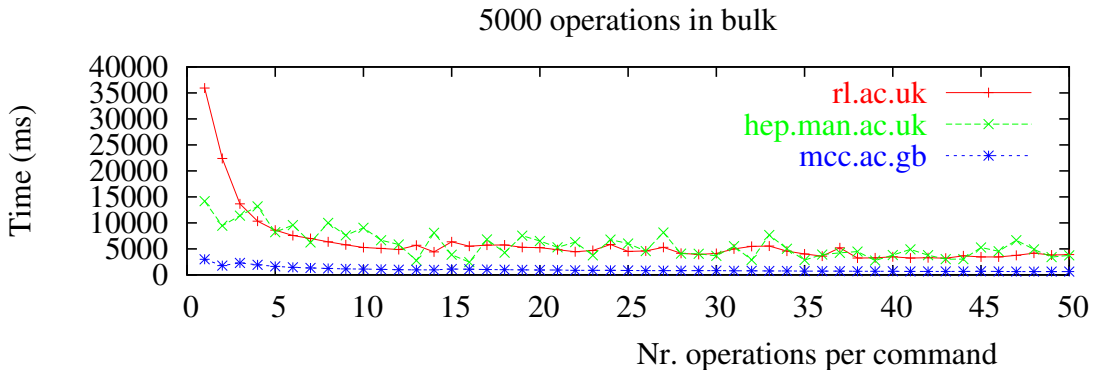


Figure D.5: `afsfsperf` encrypted bulk operations

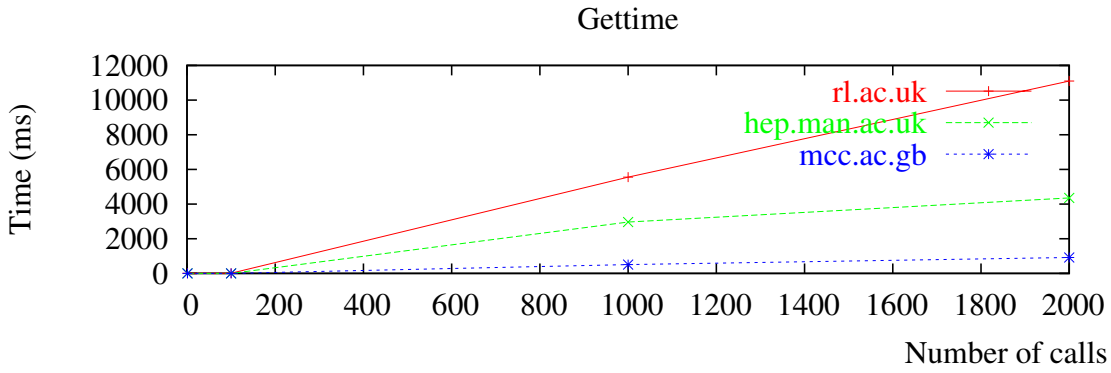


Figure D.6: `afsfsperf` unencrypted unauthenticated time requests

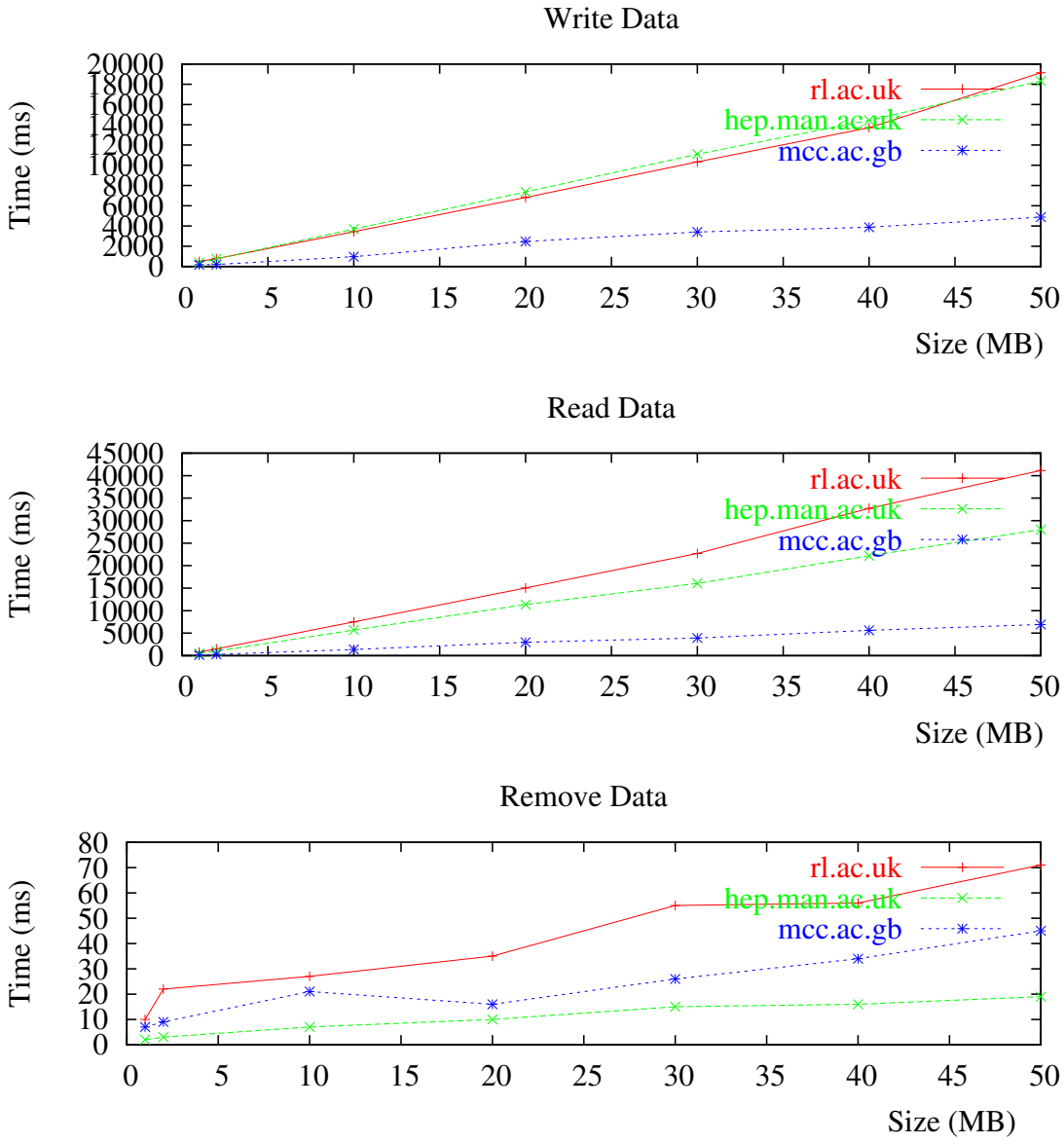


Figure D.7: `afsfsperf` unencrypted unauthenticated data measurements

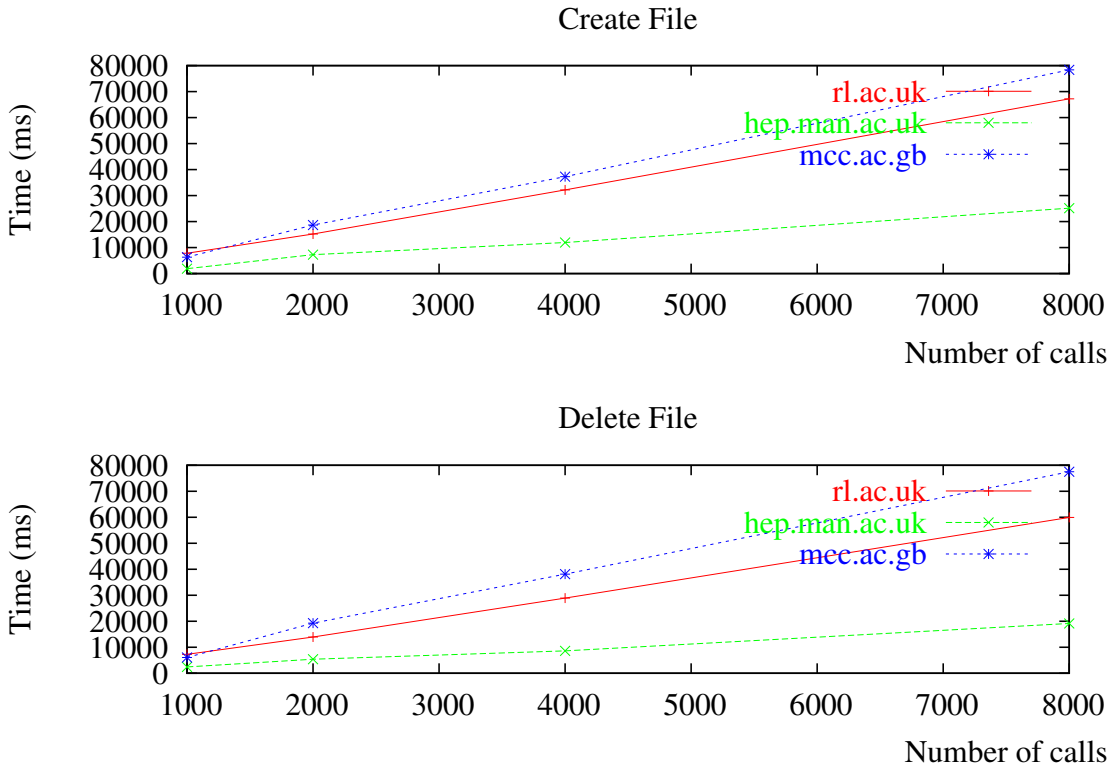


Figure D.8: `afsfsperf` unencrypted unauthenticated file operations

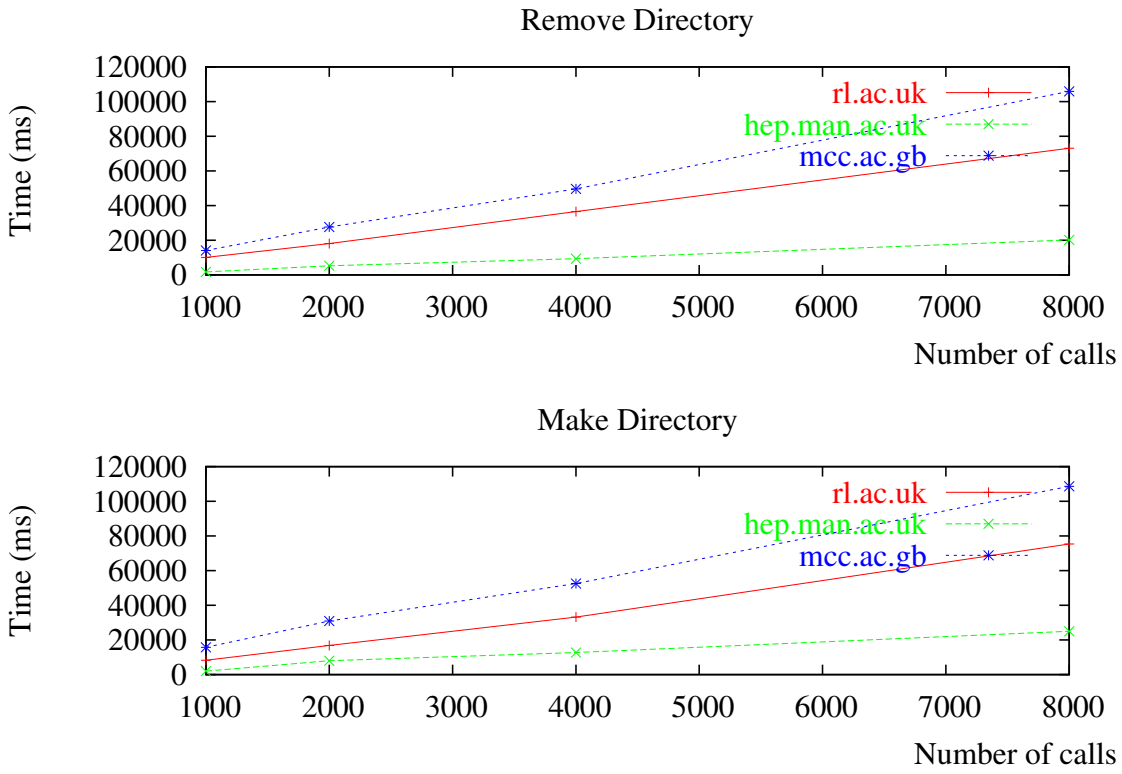


Figure D.9: `afsfsperf` unencrypted unauthenticated directory operations

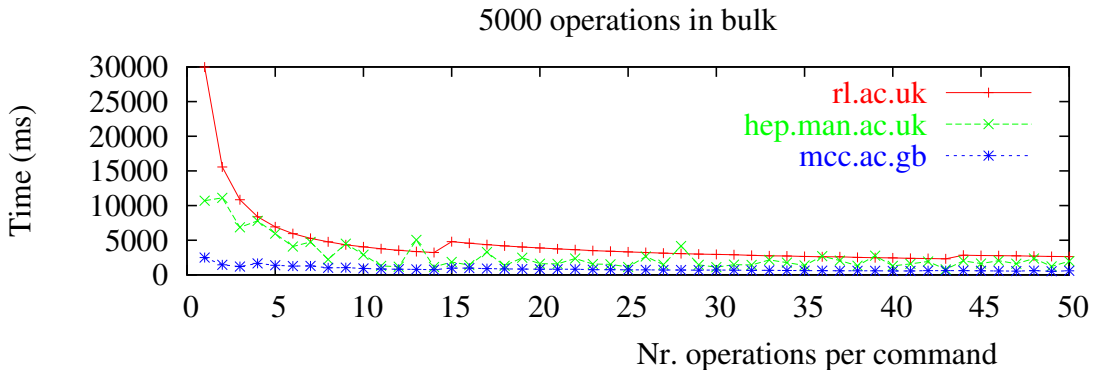


Figure D.10: `afsfsperf` unencrypted unauthenticated bulk operations