Supporting the challenge of LHC produced data with ScotGrid

Alasdair David Earl

Thesis submitted for the degree of Doctor of Philosophy
The University of Edinburgh

For every complex problem there is an answer that is clear, simple, and wrong. *H L Mencken*

Abstract

Data management is a significant research area which encompasses hardware, software and knowledge specific to the problem-domain being addressed. Particle physics is a subject which provides data intensive scenarios at the Peta-scale level, primarily the experiments using the CERN Large Hadron Collider (LHC) which is expected to come online in 2007. Current designs and simulations suggest it will produce data in excess of 15 PB / year. We are therefore, challenged to provide new ways to catalogue, store, locate and retrieve this in a timely manner.

In this thesis I, with assistance from various research groups, investigate the issues particle physics confronts in terms of data management between primary and tertiary sites when moving the computing architecture of an existing production experiment (BaBar) to the Grid.

The development of the ScotGrid prototype regional / Tier 2 computing centre is discussed from its beginnings to production. Through this I apply the experience from BaBar to a future experiment (LHCb).

I also investigate the application of storage management software as a problem for Tier 2 sites in terms of conflicting Grid interfaces and how this subject can be addressed and the progress made in its solution.

Acknowledgements

Family and friends have been an important encouragement to completion, especially Rachel who has spent the last year forcing me to write when I haven't wanted to.

My supervisor, Steve Playfer, has been a source of encouragement for this, the first e-Science PhD, and I am grateful for all his help.

Thanks to Akram Khan, Adil Hasan and Dominic Boutigany, especially in relation to the BaBar chapter, Steve Thorn for the ScotGrid chapter and the various people at SDSC, LBNL, RAL and CERN who I have worked with in the areas of SRB and SRM for Chapter 6.

I would also like to thank Anna Kenway and Malcom Atkinson, as well as the staff at the National eScience Centre, for their support and encouragement as one of their most frequent users, visitors and, I'm sure, funding sinks.

Finally I would like to thank the Particle Physics and Astronomy Research Council for the studentship which funded the first three years of my PhD.

Declaration

I declare that this thesis has been written entirely by me. It has not been submitted for any other degree or professional qualification. Except where stated, all the work detailed in this thesis was carried out by me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the recurring problems of supporting scientific advancement from a historical perspective, our motivation for investigating data management systems, and their relationship to the Grid and a road map to this thesis.

1.1 Research tools for the natural sciences

One of the new scientific disciplines to emerge at the end of the 20th Century, e-Science, is concerned with understanding how its practitioners can support computationally demanding scientific research and develop an infrastructure to enable this. To help us understand what e-Science is trying to achieve we first consider a previous period of rapid development, the late 19th Century, where both industry and academia went through great changes. This helps us to understand how we could focus our current research for maximum effect, and what problems we may potentially face.

Data management technologies that are used to record, index and distribute data and are at the heart of this thesis. These were primitive at the start of the 19th century, and the centuries old technologies being used were no longer suited to the problems they faced. Libraries had provided the primary means of data storage for thousands of years, but they were limited, as each one catalogued information in a unique and linear way.

The Dewey Decimal Classification system [106], developed in the 1870s, gave researchers a way to index their information in a standard format. This improved the

researchers' ability to group data through a common nomenclature and to rapidly locate records in unfamiliar libraries. It did not however, allow them to rapidly re-catalogue data. This forced them to create their own links between works which contained data in fields tangental to the item's core area.

Global communications networks were created by many of the imperial powers to politically control wide areas. They allowed industry to develop new opportunities by increasing trade. They also gave researchers access to new data in the natural sciences. Scotland was at the forefront of these developments, both inventing, and investing in, the technologies which gave us an international communications network and a significant repository of data in the natural sciences. Research in areas such as communications and undersea cables [12] allowed the UK to be the first to provide transatlantic communications. This provided significant advantages for scientific research at the start of the 20th Century.

The late 20th and early 21st centuries have many similarities with the late 19th century. The equipment available to scientists is going through a period of rapid development which is straining the capabilities of current technologies to catalogue, distribute and analyse data. At the same time, computing advances are reducing the barriers to entry in some disciplines, and increasing the ability of non-scientists and 'amateurs' to become involved with cutting edge scientific developments such as Seti@Home and Cancer-Busters [71, 14, 26, 16].

The current leap in technology began with the introduction of mechanical calculators and digital computers in the 1940s and '50s. These early machines allowed scientists to conduct calculations at a rate previously unimaginable by automating tasks. At the time some people questioned the need for such systems but history has shown that scientists are able to make use of whatever resources are available. This massive increase in calculating power has allowed scientists to strive for ever greater refinement in their models and use much larger data sets.

Most data is now stored on networked archival systems instead of books and journals. Like the libraries of the 19th Century these are only now starting to agree on standardised catalogue systems, such as Dublin Core[1] [22], after years of incom-

[1]Dublin Core is a metadata standard for archival systems developed internationally during the 1990's to aid the exchange of long term records.

patibilities. Data transfer has been aided by advances in fibre optics, with latency reductions and bandwidth expansion. Our ability to lay undersea cables around the world, an industry still dominated by British firms [13] has advanced communications through the use of the Internet and the World Wide Web.

1.2 Motivation

Data handling and management is an interesting issue which is often overlooked or, when remembered, misunderstood. In the attempt to advance science we often regard the ability to process data and to transport it around the world as being the whole issue. However, all this data we are creating is useless unless we can store, sort and manipulate it. In this thesis we investigate the role data management systems play in the Grid as part of the fabric for particle physics experiments. These have a bearing on all aspects of e-Science and Grid computing as they are the bottleneck in the majority of systems. Improvements in our data handling capabilities will allow particle physics to take advantage of the potential of Grid computing.

1.3 Thesis structure

Chapter 2 will introduce e-Science and the computing models it currently encompasses. It provides a description of the computing infrastructure used for the research described in this thesis, and concludes with references to other Grid infrastructures which are being developed outside the Large Hadron Collider (LHC) experiments. Chapter 3 provides details on the theories e-Science extends, which the reader will require to clarify the concepts outlined in Chapter 2, and to understand the research in the subsequent chapters.

Chapter 4 starts the description of the novel research undertaken for this thesis with an analysis of the BaBar experiment's computing model. It discusses how an active experiments may be migrated to an e-Science and specifically Grid computing framework. We provide information on the work done to aid this migration using the BdbServer++ project.

Chapter 5 presents an assessment of the ScotGrid prototype regional computing centre for LHC. We define its rôle within the current LHC computing framework and

discuss its capabilities, limitations and uses. The results from this assessment are presented in Appendices A and B.

Chapter 6 presents the SRM2SRB project. This project will enhance data management capabilities. Chapter 7 summarises the conclusions reached from this research and makes predictions where current research is likely to lead, and suggestions for future investigation.

1.4 Text Conventions

To aid the reader we highlight key words and phrases which are described in the glossary. This format will be used at the first instance of the word or phrase in a chapter and, if clarity is necessary again in the section or paragraph it appears in. Further references are provided by reference number and can be found at the end of the thesis.

Chapter 2

e-Science

This chapter provides the reader with an insight into what e-Science and Grid computing are, and why they are important tools for research in the natural sciences. Relevant development work conducted during the last decade will be discussed, and references to sources of further information are provided. This development work is focused on providing the computational and data management infrastructure necessary for the Large Hadron Collider (LHC) at CERN.

We describe projects being undertaken at the local, national and international level to provide computing and data handling capabilities for the LHC. We present some existing projects which are making use of the capabilities being developed. Some alternative tools are presented as a contrast to the approach currently being taken.

2.1 What is e-Science?

e-Science is defined as

> The common electronic infrastructure that research uses to support data and computationally intensive science

Its practitioners are developing generic tools which will allow scientists to take advantage of massive quantities of computing resources, distributed data sets and the network infrastructure which connects them. It is hoped that e-Scientists will be able to develop far more powerful software than is currently available, leading to improved quality of results, and ensuring that significant research is neither lost nor forgotten.

While e-Science does not dictate how developers provide these capabilities, there have been three areas of significant development in the last decade - Grid Computing, Web Services and Peer-to-Peer (P2P) systems. Two of these, Grid Computing and Web Services, have a direct relevance to the particle physics projects of interest to us. In the next few sections we provide an outline of these areas, leading to a more technical discussion in Chapter 3.

2.2 Computing Models

We present some computing models which have influenced the evolution of Grid computing, and which are used as resources that are available on the Grid today. This is not an exhaustive review, but provides a flavour of what the Grid, Web Services and P2P systems developed from. This will help to understand what problems they each aim to address.

Metacomputing [105, 126] research has been conducted since the 1960's to provide systems with features such as single sign-on, resource management, etc. Many of these concepts have been used in the Grid, as several early developers had previously been involved with metacomputing initiatives. Interestingly, metacomputing intended to present dispersed resources through a single interface to the user. P2P networks have used this approach, while the Grid provides multiple entry points and interfaces depending on the task being undertaken. Typically, these systems have been developed to address problems where distributed groups of users wish to make use of specific scarce resources such as experimental equipment.

Distributed systems [19, 53, 50, 54] have been developed over many years using several paradigms including distributed components, distributed objects, and remote procedure calls. They share the basic premise that specialised facilities and services can be provided over a network, with a high enough level of reliability and performance, such that the user is unaware that the majority of the system is not locally based. The BaBar experiment, which we present in Chapter 4, has used distributed systems to great effect for their computing needs.

Supercomputers [40, 20, 42] evolved from mainframe systems to provide high performance data throughput capabilities in a local system. Supercomputers, such as the HPCx facility at the Daresbury Laboratory shown in Figure 2.1, provide low latency

communication between subsystems. They are ideal for closely coupled problems such as Lattice QCD calculations and weather modelling. They differ from the other systems discussed, as they have a single administrative domain, and are intended to process specific problems sets for which the machine is designed to be optimal. The Grid treats these machines as network attached resources much like experimental detectors and other hardware.

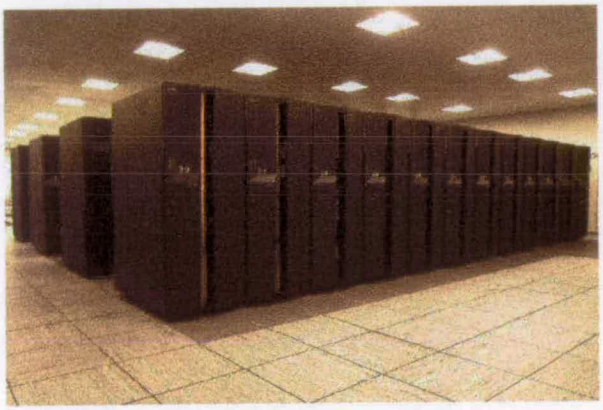

Figure 2.1: The HPCx Supercomputer

High Performance Computing (HPC) sits on the middle ground between supercomputers and commodity machines. HPC systems provide facilities, such as low latency communications or large computational capacity, through a mixture of commodity and bespoke hardware and software. These systems can be specialised for a particular problem, or generalised to provide services for a wide number of applications.

2.3 Grid Computing

Grid computing evolved from within the scientific and research communities, as the models in the previous section proved to lack some of the capabilities required. The term *Grid* was chosen as an explicit parallel to the electricity grid. This metaphor was chosen as it was assumed that, like the power grid, there were a small number of large resource providers who service a much larger number of users through a standard interface. However, like the electricity grid, the Grid has discovered that the centralised model is flawed and that, while there are still large providers, niches exist to be filled by groups with specialist requirements and capabilities.

The particle physics community has been one of the most active adopters and developers of the Grid as access to computing capabilities, equipment and data storage for widely distributed users is needed, as we will show in later chapters. Initially, particle physics used the Grid to provide access to computing capabilities and data resources. The remit of the Grid has since expanded to include improving the use of resources. This is done through optimising the discovery of available resources, providing access to these resources, and managing data replication more efficiently.

Grid computing builds on the research which has been conducted [126, 59, 105] in the area of organisational management. This studies how new communications mediums, increased collaboration between companies and countries, and the transient nature of collaborations can be formally identified and catalogued. One of these newly proposed organisation structures was Virtual Organisations (VO). VOs formed the theoretical basis for the early development of the Grid. Scientific collaborations fulfil the definition of Virtual Organisations almost perfectly [99, 124] as we shall show in Section 3.1.

2.3.1 The Globus Project

Globus is a Grid computing research and development project. Its initial research team, at Argonne National Laboratory, the University of Chicago and the University of Southern California Information Sciences Institute (USC ISI) has recently been expanded with the creation of the Globus Alliance. This brings into the collaboration the University of Edinburgh and the Swedish Centre for Parallel Computers. The Globus Project and the toolkits it has released are discussed in detail in Section 3.5. This section will provide some history of the various toolkits which have been released by the Globus team, at a level appropriate for the general reader.

The Globus Project developed out of a demonstration project for the SuperComputing '95 conference called the I-Way. This metacomputer connected 17 sites over 11 networks to demonstrate the feasibility of running applications over multiple, geographically distributed high performance and supercomputer systems. The I-SOFT software package provided the programming interface and basic software services of the I-WAY. 60 applications were developed using it and presented at the conference. After the success of the demonstration, I-SOFT was funded as the Globus Project.

The first public release of the Globus toolkit was aimed primarily at US supercomputing sites, government agencies and research groups. NASA adopted it for an Information Grid and it was used by various other projects to provide distributed access to processing capabilities. In the summer of 2000 the work of the Globus Project was presented at CERN. It met a community who were looking for an improved paradigm for the LHC computing challenge compared to the hierarchical model then in use. The adoption of the Grid by CERN underpins the relevance and need for the work presented in this thesis.

The second version of the toolkit [99, 124] provided a much improved installation interface and a new structural model based on three pillars (see Figure 2.2) comprising resource management, information management and data management. This toolkit was taken up by the physics community in great numbers on both sides of the Atlantic and formed the basis of the European DataGrid. At this stage many of the scalability issues of the system were noticed, and a great deal of development work went into providing solutions to these problems.

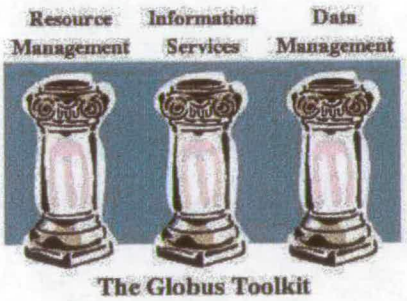

Figure 2.2: The Globus Version 2 pillar model

The planned development of version three (GT3) of the toolkit was announced in February 2002 at the Global Grid Forum (Section 2.3.3). This introduced Web Services (see Section 2.4) to the Grid community, and defined the requirements for a Grid Web Services Description Language (GWSDL) [100]. This specification extended Web Services by defining a persistent service, known as a factory [142]. These factories could be used to provide functionality similar to those in use on the Unix-based Grids then available.

The use of a Web Services framework overcame some of the scalability issues which earlier versions of the Globus software had faced. However, moving the model the

Grid community was using from homogeneous Unix-based systems to a heterogeneous one has added complications. A further issues has been the design of Web Services. These were created with reference to a very different problem domain than the particle physics community. Web Services will be discussed in more detail in Sections 2.4 and 3.6.

Version 4 of the Globus toolkit was announced in January 2004 at the GlobusWorld '04 conference and is due to be released toward the end of 2004 [125] together with the WSDL v.2 standard. It will incorporate the Grid community's definition of a factory as a persistent web service, which should bring the Grid into line with mainstream Web Services, allowing a greater interchange of software between industry and research.

2.3.2 Unicore

Globus is not the only basic toolkit for Grid computing that is available today. The UNiform Interface to COmputing REsources (UNICORE) [80, 101] project has been developed in Germany and the UK to provide a graphic interface to distributed resources. It allows the user to specify work flows using their resources through a user friendly environment. Unicore is primarily aimed at the bio-molecular science, meteorology and computer aided engineering (CAE) communities. It has users in Europe and Japan [140]. Unicore is not being widely deployed for particle physics experiments in the UK so it will not be discussed further in this thesis.

2.3.3 The Global Grid Forum

The Global Grid Forum (GGF) [31] provides the means for research and industry to discuss the Grid at its tri-annual conferences. It aims to perform a role similar to the World Wide Web Consortium (W3C) in developing standards and providing reference implementations which are then submitted to bodies such as the IETF for general acceptance.

The work of the GGF is separated into roughly seven areas which comprise various Research and Working Groups [1]. These areas are Applications and Programming Models Environments (APME), Architecture (ARCH), Data (DATA), Grid Security

[1] Given the fast moving nature of the Grid these change on a meeting by meeting basis around a core of about 7 areas. Groups may move between them to find an appropriate fit.

(GRID SEC), Information Systems and Performance (ISP), Peer-to-Peer (P2P), and Scheduling and Resource Management (SRM).

Research Groups look at the longer term issues of interest to the Grid, and attempt to understand the direction and needs of the community. Working Groups have shorter lifespans, typically under 2 years. They develop reference implementations and define specific standards.

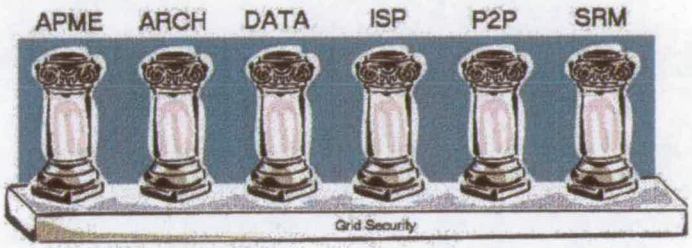

Figure 2.3: The main GGF Areas

GGF is at the forefront of attempts to merge aspects of the Grid, Web Services and Peer-to-Peer systems, where researchers have discovered common issues and requirements. These communities see a need for common research collaborations, even if the eventual solutions have a different focus.

2.4 Web Services

Web Services have been under development by industry since the mid 1990's with the term "web services" first used by Microsoft in 2000. They provide a structure for users to locate and assess capabilities, and to communicate with services using three core technologies: SOAP, WSDL and UDDI. We will discuss the technical details of these in Chapter 3. Put simply: UDDI provides a system to locate web services; WSDL provides a description in XML of the capabilities and limitations of a Web Service; and SOAP provides a protocol with which to place requests and receive responses. As such, web services provide an alternative paradigm to Grid computing for e-Science to consider when developing solutions.

Web Services offer many advantages to the particle physics community for a variety of the problems they face. However, they may not necessarily help with some of the high performance issues which need to be addressed. It is widely accepted that

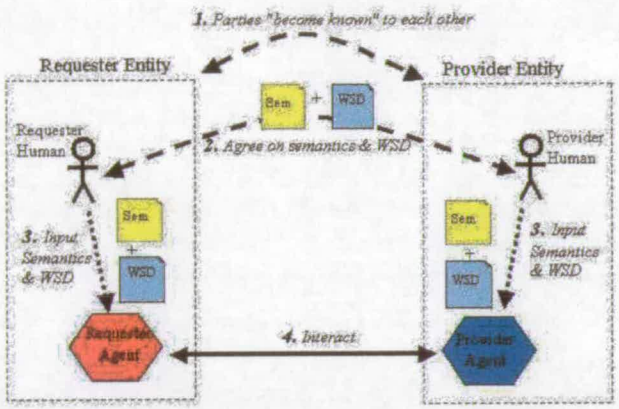

Figure 2.4: W3C diagram of web services interaction

the Grid and Web Services communities are starting to converge and that they will become more relevant to particle physics in the future. This will be discussed in relation to our work in the area of storage management in Chapter 6.

2.5 Peer-to-Peer Systems

Peer-to-Peer systems (P2P) share many of the same aims as Grid computing. The two communities are starting to talk to each other and conduct joint research in the Global Grid Forum (Section 2.3.3) to solve common problems. While Grid computing recognises that there can be a hierarchical structure in the system and that systems are not homogeneous, P2P systems typically assume lowest common denominator nodes and attempt to create their own virtual network on top of the physical one. This leads to significant performance and cost issues with the network.

There are other areas of significant difference between the two groups. The user community served by the Grid is several orders of magnitude smaller than the P2P community, but deals with similar quantities of data and resources [137, 138]. In terms of security the P2P community is primarily interested in ensuring the anonymity of users, while the Grid community is more interested in the authentication of users and the provenance and durability of data. We provide further technical information about P2P systems in Section 3.4 as some of their research and solutions are of practical value to our research and provide a useful comparison.

2.6 The CERN Perspective

There is a strong drive in Europe [23] by national governments and the European Union to ensure that the computing infrastructure required for the Large Hadron Collider is in place and working before its start date in 2007. The funding agencies also hope that this investment can be re-used for other projects, specifically those that are expected to have significant computing requirements in the future - such as medicine, engineering and earth sciences. Thanks to this, the level of European Grid deployment is more advanced than in other regions, and a large quantity of expertise and support is now available.

The Grid has proved beneficial for smaller science and engineering projects as they have been able to take advantage of the computing capacity to further their own research. Their involvement has assisted in the testing and deployment of software for the primary task, the LHC experiments.

2.6.1 The European DataGrid

The European DataGrid (EDG) project [102, 24] ran between 2001 and 2004. It had the dual role of basic development, and overseeing the roll out and acceptance testing of a DataGrid for European researchers. This work was led by CERN with the assistance and funding of the European research agencies; the European Space Agency (ESA); France's Centre National de la Recherche Scientifique (CNRS); the Italian Istituto Nazionale de Fisica Nucleare (INFN); the Dutch National Institute for Nuclear Physics and High Energy Physics (NIKHEF); and the UK's Particle Physics and Astronomy Research Council (PPARC).

EDG was split into twelve Work Packages (WPs) grouped around four areas. There were seven WPs involved with developing and deploying software, three WPs involved in application development to run on the DataGrid, and two WPs to disseminate information and manage the project. We briefly describe each of these to give a sense of the scope of the project, and discuss how these groups interacted both computationally and as an administrative organisation.

Work Package 1 [85] dealt with Workload Management. Workload management attempts to optimally match users' computational and data requirements to the available resources. This is based on who the user is, what resources their group has

Figure 2.5: Organisation of the European DataGrid Work Packages

access to, and the priority of the request. This is done through the use of a Resource Broker (RB). This is similar to UDDI for Web Services (Section 2.4). The RB can have a profound effect on the users perception of the system and the utilisation rates which can be achieved. Significant work has been done to optimise the assignment of request and ensure the robustness of the system.

Work Package 2 [88] was in charge of Data Management issues. The group looked at how users can access, store and retrieve data through the Grid with the emphasis on very large data sets. Improvements in the algorithms used for data location can have a serious impact on the retrieval costs. Likewise, the optimisation strategy used to replicate the data can also have a major impact on the retrieval costs when physicists want to run an analysis [61, 67]. Much of the novel work we present in Chapters 4, 5 and 6 is in the Work Package 2 area.

Work Package 3 [90] - Information and Monitoring Services. This looked at how to report hardware performance and feed this information into the WP1 Resource Broker and various higher level applications. WP3 provided developers with the data necessary for simulating the Grid, allowing performance tuning and the identification of problems which require human intervention. WP3 builds on the work done by WP4 but looks at the Grid as a whole rather than at a specific site. This

work draws on research with WSDL for Web Services, and the ClassAds language used by the Condor project.

Work Package 4 [91] developed Fabric Management. For EDG, fabric is the large computer clusters and storage resources at the national and regional centres and the networking infrastructure which connects them. These perform the majority of the computational work for physics. However, managing these large systems is an unenviable task, as there is a limit to the number of support people available. For LHC to be successful it needs each of the system administrators to be as productive as possible, so WP4 attempted to automate as many functions as possible from software installation, to maintenance and monitoring. The software from WP4 regards each site as individual, with their own requirements and customisations. It feeds information about the systems into WP3, which reports aggregated data to the operations centres.

Work Package 5 [92] studied Mass Storage Management. The software they developed, the Storage Element, provides an interface between the Grid and various online, nearline and offline data storage systems. This WP has a significant bearing on this thesis and we will provide information on our work in this field in Chapter 6.

Work Package 6 [93] managed the deployment of testbed systems. These provide feedback to the other work packages on installation and distributed system management issues. WP6 did not develop any software itself, but provided quality assurance and suggested solutions to ensure the systems it deployed were operational and fulfilled the challenges they faced.

Work Package 7 [94] developed Network Services. These monitor the state of the network, and assist applications and the DataGrid in routing data in the most efficient way.

There were three main application development areas for EDG. Work Package 8 [95] dealt with high energy physics applications with a strong emphasis on LHC computing requirements. Work Package 9 [96] dealt with earth observation science problems such as climate modelling, satellite systems and information gathering. Bio-Informatics was covered by Work Package 10 [86].

Work Package 11 , Dissemination, provided a public interface to the DataGrid for industry, government and the media. Work Package 12 [87], Project Management, ensured that the necessary framework was in place to plan and complete the requirements on time. The final review of EDG states that they were achieved [21].

The EDG has now evolved into two separate projects, the LHC Computing Grid [48] and the Enabling Grid for European e-Science [25].

2.6.2 LHC Computing Grid

The LHC Computing Grid (LCG) is tasked with having a Grid computing infrastructure in place prior to the start of the LHC, estimated to be in 2007. It builds on the work done by the European DataGrid with the more focused priority of providing a distributed computing system capable of supporting the simulation, processing and analysis needs of all four of the LHC experiments: ALICE, ATLAS, CMS and LHCb. This is being done in conjunction with the various national Grid projects currently under development.

2.6.3 Enabling Grid for European e-Science

The Enabling Grid for European e-Science (EGEE) [25] is funded by the European Union under Framework 6, with the aim of integrating existing national Grid projects connected by the GEANT research network [29]. The primary aim of EGEE is to support the LCG and Biomedical Grids in Europe. The secondary aims include assisting in the development of production quality Grids and in providing training [56] for researchers and students in developing their research using the Grid.

EGEE is assisting in the development of production quality Grids, which can be relied upon for applications in areas such as medicine. This requires improving security and code reliability and, if necessary, refactoring code from testbed Grids such as EDG and LCG. It has a wider mandate than the LCG, which is only concerned with particle physics, and focuses on more generic features which can be reused by many disciplines.

2.6.4 GridPP

GridPP is an umbrella organisation connecting the particle physics and computing researchers, funded by PPARC, whose primary aim is to develop the UK's contribution to the processing and storage needs of the LHC. As a secondary objective they are developing and deploying Grid technologies to assist smaller projects [55].

Figure 2.6: GridPP participating sites

GridPP has roughly 80 researchers based at 18 universities as well as Rutherford Appleton Laboratory (RAL) and CERN. GridPP ensures that the replication of effort is minimised within the UK, and that UK researchers are aware of international research being undertaken. It also ensures that developers are aware of the progress of other projects, and provides a way to solve common problems quickly through support and documentation.

There are five main computing facilities available through GridPP, the Tier 1 centre at RAL and four Tier 2 centres comprising the London, Northern, Southern and Scottish Grids. Each of these regional Tier 2 centres links resources at several universities. Of these ScotGrid is relevant to this thesis, and will be discussed in the next section and Chapter 5.

ScotGrid

ScotGrid is a joint project between the Universities of Edinburgh, Glasgow and Durham to form a Scottish regional centre for the LHC Computing Grid as part of the GridPP collaboration. The Edinburgh ScotGrid component has participants from the School of Infomatics and the School of Physics. Glasgow participants are the Departments of Computing Sciences and Physics. The University of Durham is represented by the Department of Physics.

ScotGrid has a significant quantity of computer hardware, mainly funded by the Scottish Higher Education Funding Council (SHEFC), which is split between the computation capabilities at Glasgow and Durham, and mass data storage at Edinburgh. All three sites are connected by the SuperJanet network.

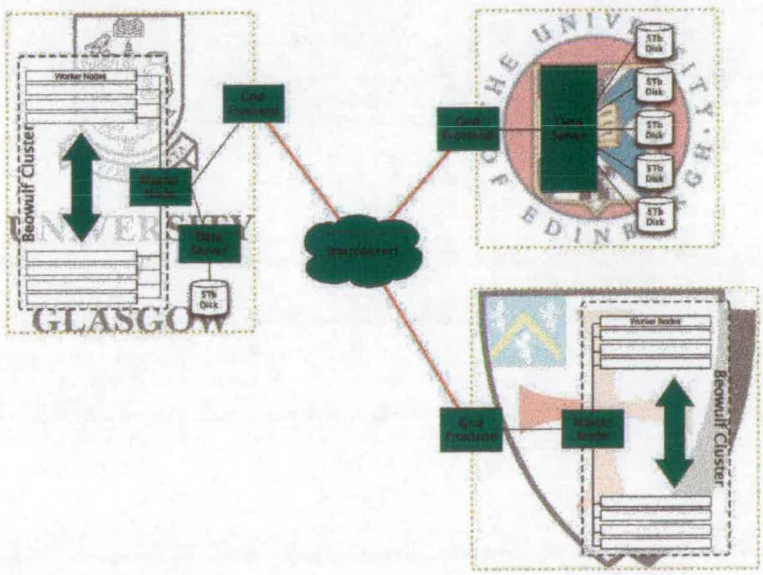

Figure 2.7: A logical view of ScotGrid

The ScotGrid project will be discussed in more detail in Chapter 5 including information about the computational capabilities, user community, and software environment. The work conducted to benchmark, provide, and expand these capabilities and assist the user community is also discussed.

2.7 Other physics grids

So far we have concentrated on the computing grids which are intended to assist physicists working on the LHC experiments. However, it is worth noting that there are many other projects which have similar aims within their own countries and collaborations. We present short descriptions of several projects at national and international levels which are attempting to provide global grid infrastructure.

2.7.1 AstroGrid

AstroGrid [2] is a sister project to GridPP, also funded by PPARC. Its aim is to support the UK astrophysics community by providing remote access to resources and data as part of the global *Virtual Observatory*. AstroGrid shares similar problems to GridPP, with its users requiring access to very large data sets. AstroGrid researchers are working on improving the methods used to locate and retrieve data from heterogeneous databases in association with the National e-Science Centre and various other groups. This will result in more data being available for the community to analyse, without requiring more telescopes, or more use of existing ones.

2.7.2 NorduGrid

NorduGrid [58] is a collaboration between the Scandinavian countries to provide a common Grid for their researchers. This was started as the Nordic Testbed for Wide Area Computing and Data Handling and became NorduGrid as part of the EU DataGrid and later the LHC computing challenge. Its intentions and scope are similar to the GridPP and AstroGrid projects in the United Kingdom. It has acquired a strong reputation for the robustness of its software.

2.7.3 Particle Physics Data Grid (PPDG)

The Particle Physics Data Grid [62] was started in 1999 with United States Department of Energy (DOE) funding [102], to act as a collaboration between software developers, such as the SDSC Storage Resource Broker and Condor teams; collaborations, such as Atlas and CMS; and laboratories including Jefferson Laboratory, FermiLab and SLAC. This is a higher level collaboration than many in the UK and would be one level above the GridPP and AstroGrid projects. It shares a similar scope to the EGEE collaboration in Europe but with a greater focus on the particle

physics domain.

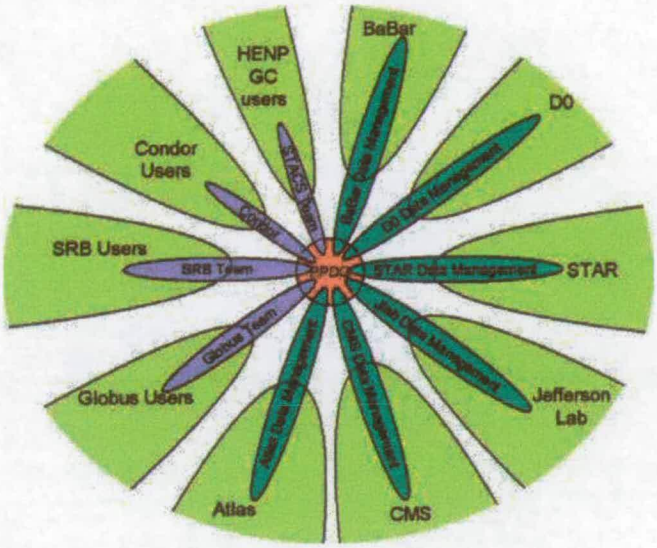

Figure 2.8: PPDG community

2.7.4 Grid Physics Network (GriPhyN)

The Grid Physics Network [35] is a collaboration between ATLAS and CMS, the Laser Interferometer Gravitational-wave Observatory (LIGO) and the Sloan Digital Sky Survey (SDSS) in the United States, funded by the NSF [102]. All of these project have a common issue with massive, distributed, datasets at the Petabyte (10^{15} bytes) scale. As a result they have pooled their research in DataGrids. GriPhyN focuses their research effort on common problems such as data management techniques, scheduling, and transaction management.

2.7.5 BaBarGrid

BaBarGrid is an informal group of people within the BaBar collaboration who are working on the development and deployment of Grid applications for the experiment. BaBarGrid will be discussed in more detail in Chapter 4 but is summarised here to aid the reader.

BaBarGrid has a significant European contingent which participated in the development and deployment of the European DataGrid. This resulted in facilities such as Resource Brokers (RB), Virtual Organisation management systems and Replica

Catalogues (RC), being deployed in the context of the BaBar computing model. It has continued this relationship with the LCG but faces the obstacle that it is a currently active experiment, and cannot change its computing model to an unproven one at this time.

2.7.6 International Virtual DataGrid Laboratory

The iVDGL [102, 44] is involved with developing a Grid for data intensive research in the physics and astronomy domains. This is intended to build on the work of globally distributed groups such as PPDG, GriPhyN, the EDG and its successors to make them compatible with each other. Figure 2.9 shows the network links between the sites involved with this. It has an Operations Centre located at Indiana University which provides information on status, configuration, management and robustness of the sites.

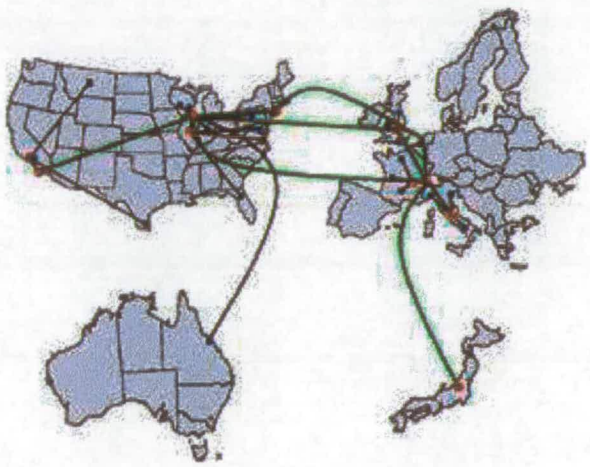

Figure 2.9: iVDGL participating sites

2.8 Conclusion

In this chapter we have presented our understanding of e-Science, its aims, foundations and an overview of the technologies which can be used to implement it. From these overviews we can see clear parallels between each of the approaches, in terms of their need to address processing, networking, data management and information publication and discovery systems. We present some of the technical details of how

these issues are being addressed in the next chapter.

With the limited detail which we have presented so far about the computing infrastructure being developed for the LHC, it should be obvious that to succeed, collaboration between nations and universities is necessary. Projects at the regional, national and international level have their place and there is the potential for significant efficiency gains if managed correctly. We discuss the gains which can be achieved at a regional level as part of Chapter 5 in relation to the ScotGrid project.

Chapter 3

Concepts for e-Science

e-Science and the implementations it builds on, Grid computing, Web Services and P2P networks, draw on many areas of research from computer science and elsewhere. This is especially true in the areas of security, organisational theory, distributed computing models, network transport protocols, web technologies and operating system concepts.

In this chapter we outline the theoretical work and concepts behind virtual organisations, and how this concept is applicable to e-Science and the physics experiments we are interested in. We discuss the computing infrastructure which e-Science needs to develop to support research in terms of distributed systems, and specifically data management, which will be developed in later chapters. We look at which aspects have been emphasised by the Grid, and the evolution it has experienced from Unix/Linux based software through web services, to the current Grid Services model.

3.1 Virtual Organisations

Virtual Organisations [124] were described by management researchers based on the real world experiences of industrial consortia and academic research collaborations. These organisations are characterised by having dynamic collections of individuals and groups which require flexible, secure, co-ordinated systems of resource sharing to achieve a common aim. These collaborations can vary tremendously in the scope of their purpose, size, duration, structure, community and sociology.

We provide three examples of existing virtual organisations, spanning an interna-

tional collaboration, a regional collaboration which is part of several larger ones, and a local collaboration. These present a simplified view of the issues and complexities faced when defining a VO, and the problems faced when attempting to draw clear lines of separation between groups which share resources and/or personnel.

3.1.1 The LHC Experiments

The LHC experiments will generate at least three orders of magnitude more data than any prior particle physics experiment. The number of active collaborators will be ten times those of the Large Electron Positron (LEP) collider experiments. They have computational requirements at the very limits of what existing systems and techniques are capable of providing.

Each of the four major experiments associated with the LHC: ALICE [1], ATLAS [3], CMS [18] and LHCb [49] is an example of an interesting and highly dynamic Virtual Organisation. The collaboration attached to each experiment will evolve significantly over the project lifetime, and the range and constituency of membership and resources will change drastically. In Chapter 4 we discuss how the BaBar collaboration's computing infrastructure changed over its lifetime to provide an example from an active experiment.

For each of these experiments the core purpose of the VO is to enable researchers to access data produced by the experiment to perform analysis work. Theoretically all data from the detector and simulation production could be stored at a central site, however, from experience with other experiments we can see the advantages, in terms of cost and performance, that distributed systems can provide. There are many secondary purposes for the virtual organisations, including more efficient use of resources, productivity increases through the automation of tasks, and focusing of expertise.

The membership of each experiment's VO is dynamic, as each experiment will have a life-time of around 20 years, from inception to final analysis. During this time a huge number of post- and under-graduate students, researchers, staff and users will join, leave and potentially rejoin the experiment. An individual's role will change as their career progresses, for example acting as lead programmer or run coordinator, and the systems used must be able to deal with this.

At present, while the LHC is being constructed, work is being done in the areas of design, testing and compliance for the physical infrastructure, and development and testing of software for the computing requirements. New groups and individuals are added to the VO on a regular basis, as projects progress and designs and intentions change. The computing testbed aspects of this are discussed in Chapter 5.

During a physics data challenge, which is a concerted effort on the part of several sites to produce a specified amount of simulation data in a given time window, sites will make resources available either exclusively or with a higher priority, in the form of compute capabilities and human resources for the duration of the task. This may require: a reconfiguration of existing software; the installation of new software; changes to the access lists, to allow new users, or to restrict existing ones; and scheduling of other resources such as data storage.

When the LHC comes online in 2007/8 the experiments will face the security issue of ensuring that the acquired data, which is likely to be stored on the same machines used by other experiments, can only be contributed to by authorised members of the experiments data production team. Access permissions for specific files and information will change over the lifetime of the data as it is produced and researchers are granted access to it. Likewise, small groups will form to analyse specific data samples and will want to be able to securely share results, resources and data.

Access to data alone will not enable new physics results. Researchers will also require access to software to analyse the data, and computing resources to run this software on. This presents new challenges as security must be in place to regulate access and to ensure that rogue programs are not able to run rampant. New techniques will have to be developed to achieve this, some of which we will discuss in Chapter 5.

This briefly outlines, in broad strokes, the complexity of dealing with the computing requirements for the LHC experiments. The specific details of how each of the experiments is addressing their particular computing and data requirements are available from the project websites referenced at the start of this section.

3.1.2 ScotGrid

The ScotGrid [70, 121] project provides a regional computing centre for a variety of virtual organisations including LHC and US based experiments. It has participants from six institutes spanning three universities, supporting numerous users scattered around the world. Edinburgh acts as a data repository, as well as providing a small analysis facility. There are further analysis and simulation production facilities available at Durham and Glasgow. The project requires that access to resources and data be able to differentiate between local users and remote ones, and to assign priority.

Data can take many different forms in the system, from short term, temporary files to long term repository data. The ability to allow or deny individuals access to data, based on their identity, affiliation and role is therefore crucial. In the ScotGrid VO information about casual users must be updated regularly at each of the sites, information about people at the hosting sites maintained to allow priority access, and effective quotas for both processing and data storage must be enacted, while still giving flexibility for users and groups with dynamic needs.

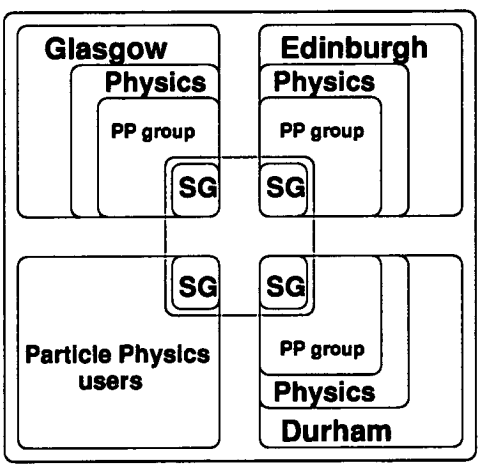

Figure 3.1: A partial view of the ScotGrid virtual organisation

In Figure 3.1 we show the ScotGrid VO as a subset of the particle physics groups at the Universities of Edinburgh, Glasgow and Durham. These groups are themselves subsets of their respective schools and departments which are subsets of the universities. The particle physics users of the system are themselves a subset of the international community of particle physics researchers.

3.1.3 Industrial Projects

Projects between universities and industry are becoming more popular and wide spread. Confidentiality is required for these projects when they have the potential for short term commercial benefit. In these collaborations the various partners may wish to grant access to specific resources to named individuals or groups. These people require some form of online authentication which does not impose an undue administrative overhead and which can be revoked or altered as necessary. Significantly these groups may be a subset of a larger organisation.

Access to data and information is a significant issue. A comprehensive audit trail may be necessary to ensure that there are no unauthorised or unwanted copies in existence, while at the same time ensuring that network overhead is minimised. As an example, a collaborative project between a university and a medical research company may be formed to allow academics to test new search algorithms on a biology database. The company has an obvious interest in greater efficiency of their algorithms, but must safeguard parts of their data to ensure competitors do not gain access to it.

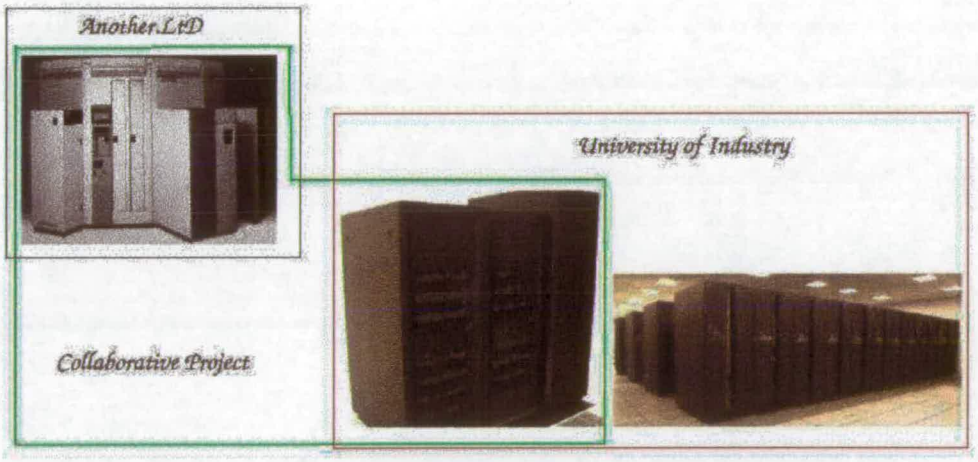

Figure 3.2: A collaborative Virtual Organisation

We show in Figure 3.2 a collaboration, labelled Collaborative Project, involving a database provided by Another LtD and a subset of the computers at the University of Industry. The collaboration uses a subset of both organisations resources to produce a result which would not be available without these facilities being used in conjunction.

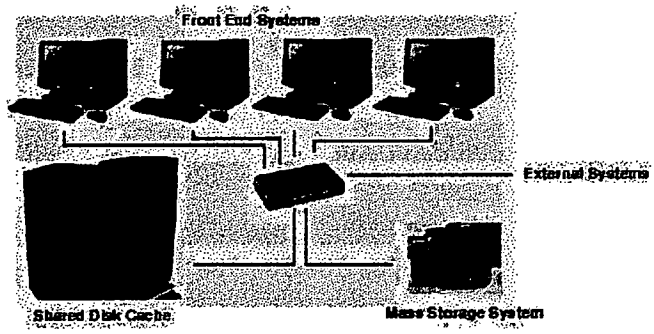

Figure 3.3: An example hierarchical storage system

tape. This uses the disk cache as a buffer to ensure the data can be re-transmitted in the event of a network failure.

3.3.2 Distributed File Systems

Distributed file systems have been an area of active research for several decades. They are concerned with many of the same problems that traditional local file system face such as the naming, creation, deletion, retrieval, modification and protection of files [108, 114] and the management of resources.

In distributed file systems the software must be able to deal with issues related to the nodes which store data, files, users, and the number of files in the system e.g. the addressing scheme. Issues such as file locking, cache consistency and version control have a number of potential solutions and implementations, which are well discussed in published literature [127, 129, 108, 114].

It should be stressed that distributed file systems are not the same as distributed data systems and are certainly not the same as DataGrids. File systems are based on files being a named collection of logically grouped data such as a document or program which is stored and retrieved using an interface which presents a traditional, centralised view of the data. Data systems, such as the Grid, are interested in dealing with data and information in much more advanced ways as we shall show in later sections.

3.3.3 Data Replication

Data in distributed environments is replicated for a variety of reasons including performance, reliability and availability [109, 115]. As part of the LHC computing challenge data sets in the Terabyte range will be common, and the latency that users would experience if data was stored at a single site, potentially very distant from the analysis site, would be unacceptable.

To deal with this work has been done as part of the EU DataGrid's Work Package 2 in terms of data optimisation [67] and data replication [89]. This group has developed the OpterSim [61] package for simulating data in Grid environments to test the effect of using different data replications strategies.

3.3.4 Data Location

Data location is a significant issue in distributed systems as the number of individual data entities grows toward the billions. To locate and retrieve data in a timely and efficient manner metadata about where files are stored, what format they are in, what security or authentication is needed to retrieve them, must be provided. This metadata requires an infrastructure to be in place for users to locate data. It must also ensure that data is maintained in a consistent manner. Data which has been moved or deleted must be marked as such in the metadata system, or users will be unable to trust the results it returns.

There are many methods of doing this which involve both the reporting mechanism, and the addressing scheme [116]. We outline a few abstract methods of achieving this here, and discuss the methods used by the Storage Resource Broker and Storage Resource Manager in Chapter 6.

To locate data we have at one extreme a single, centralised, location and a at the other a fully distributed system, with a hierarchal system taking the middle ground. Centralised systems have been used for small systems in the past - such as single machines or a local network. However, the capacity and reliability difficulties created by this model have normally forced developers towards a more distributed system.

In a distributed system, each site or node maintains information about itself locally and must link to other nodes to query whether they contain the data required. This

has the advantage that the metadata overhead at each node is a function of the quantity of data stored there, but has the disadvantage of having a high network overhead, as requests are passed back and forth in the attempt to locate data.

Hierarchical systems, such as the one adopted by the European DataGrid and LHC Computing Grid, form a halfway house between these two extremes. In these systems each branch maintains information about the contents of the leaf nodes below them so that large quantities of information can be queried quickly and without the overhead of having a single, central site.

3.3.5 Security

Security for data storage systems is a multi-layered issue which cannot be solved by one group or product alone [110, 118]. It cannot be seen as something which is an addition to existing systems, but must be considered from the outset and addressed accordingly. We identify the security aspects of storage systems as falling into four categories: access; storage; transfer and accounting. We will discuss these as part of a traditional, single site system before expanding on the issues that the grid will involve.

Access to data stored on a logically structured system must ensure that the user has the correct level of authorisation for the task they are attempting to undertake. For example, users may be allowed to perform a subset of the following actions; read, write, execute, query, copy, move, modify, store, and delete. The actions the user is allowed to perform may also be limited by how they connect to the system, what task they are performing etc. As an example a user who is a run coordinator will be allowed to add data to the system while on duty, but as a normal user they may not.

Once data is stored we must ensure that it remains uncorrupted, unchanged and untampered with. There should be a facility to discover where data came from prior to its current form. Knowledge of where data was created or received ensures that, if corrupt input is discovered, it can be located and repaired, and the source of the problem identified and fixed. Of the two areas, the latter is most difficult to address and remains an open research area. Ensuring that data is uncorrupted is achieved through the use of checksums, and following good practise for data backups.

Accounting provides a record of what has happened in a system. It potentially offers the opportunity to roll back the system to a previous stable state if necessary. A comprehensive accounting system should be able to maintain knowledge of the logical and physical information about files and cached data, information about users and access permissions, and information about the system itself.

When we move to a grid system we wish to provide access to a range of heterogeneous storage systems which may be geographically dispersed. The security issues must then consider whether we move the authentication and authorisation functions to a higher level and if so how we can address the legitimate concerns of system administrators who wish to ensure that the security system they are using is robust, and that the actions of users can be properly accounted for in the event of an attack, that the risk of corruption of data is minimised.

3.3.6 Data Transfer

Data transfer is the moving or copying of data from one system to another external one, typically via the internet. There are many points where data can become corrupted during this as it must pass through several layers of the network stack before being transported, and then be successfully reconstructed at the receiving machine.

Further security may be added to the data at this stage by the use of Transport Layer Security (TLS) to ensure that it cannot be intercepted by a man in the middle attack. Various data transfer protocols can be used to ensure optimal performance of the system and many different types of network can be used including copper, fibre and wireless.

3.4 Peer to Peer Systems

Peer to Peer (P2P) systems have proved to be a popular method of implementing distributed file systems, and are an interesting contrast to Grid systems [103, 130]. They address issues of replication, location, retrieval and resource discovery based on the assumption that they consist of a large number of unreliable nodes. One of the key points of this model it that nodes must use a registration scheme which is robust enough to deal with them joining and leaving the system on a regular basis without the system collapsing.

P2P networks create their own virtual network for routing information and data on top of the physical network. These virtual networks have significant cost and performance issues for the service providers of the physical network as they are developed from an imperfect knowledge of the physical resources and capabilities and are inefficient as a result [137, 138].

P2P networks started from a contrary position to the Grid. While the Grid has a small number of users with massive requirements, P2P has a massive number of users with small requirements. P2P networks were initially flat and unstructured while the Grid started from a hierarchical model. The Grid has decomposed its hierarchy while P2P systems have become hierarchical with the use of super-nodes and hubs.

Grid and P2P research is converging, but there are a significant number of distinct differences between the two groups. This include the technical sophistication and access to resources the groups possess. While the Grid community may be smaller, the aggregated resources of the two groups are similar. This suggests that in the course of the next few years both groups will continue to borrow ideas from each other, but there is unlikely to be a single solution which satisfies all of the requirements of both.

3.5 Globus 2.x based Grids

The early releases of the Globus Toolkit [128, 124, 99, 104], Versions 1.x and 2.x, were aimed primarily at Unix and Linux based systems, as these are commonly in use at computing centres and university research groups involved with computational and data intensive research. A significant number of Grids which are now in production use are based on these versions including the LHC Computing Grid. These Grids extend the basic functionality of the Globus Toolkit. In the next few sections we discuss the basic functionality that the Globus Toolkit provides as a reference for later development.

3.5.1 Resource Access

The Globus Toolkit provides access to resources through the use of Globus Resource Allocation Managers (GRAM). These provide the interface between the network facing Grid interface, known as the gatekeeper, and a resource on the local machine. GRAMs can fork processes into a user account or can submit jobs to a variety of batch systems including PBS, LSF, and Condor. Each gatekeeper will have one default GRAM but can access many others providing they are specified by the user or the default GRAM is capable of allocating jobs to other GRAMs.

Globus Security

Security in the Globus Toolkit [32] is provided through the Grid Security Infrastructure (GSI) package which is based on the GSSAPI. This makes use of asymmetric cryptography [133] in the form of digitally signed public private key pairs which are authenticated by a Certificate Authority (CA) and utilise the X.509 [43, 131, 43] standard. To aid later work in Chapters 4 and 5 and Appendix A, we present a distilled version of how the Globus Toolkit provides secure authentication between a client and server in the sequence diagram, Figure 3.4.

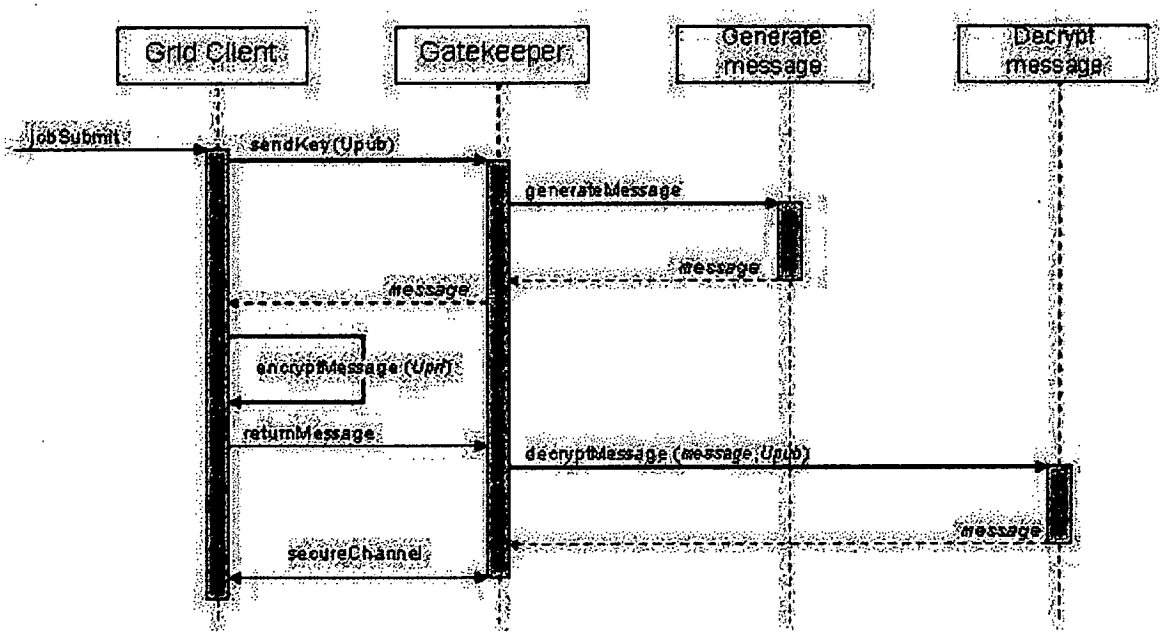

Figure 3.4: Sequence Diagram for User Authentication

The Globus Toolkit mutually authenticates the client to the server and vice versa by a relatively straightforward method. The Grid Client (GC) provides a copy of the user's public key or certificate, Upub, which has been signed by a Certificate Authority (CA) to the Gatekeeper. The Gatekeeper uses the distinguished name (DN) in the certificate to select which CAs' public key to use in testing that the users certificate is valid and has not been revoked. For simplicity this is not shown in the diagram.

Assuming that Upub passes this test, the Gatekeeper creates a randomly generated message which it passes to the Grid Client in clear-text form. GC encrypts the message using the users private key, Upri and transmits the cypher-text to the Gatekeeper. The message is decrypted using Upub and, if the clear-text matches the original message, the GC is assumed to be who they claim.

The processes would then be reversed, with the Gatekeeper transmitting its public key to the GC and the same procedure followed. Once both of these authentications have been successful a secure channel, based on symmetrical keys, would be created for further communication.

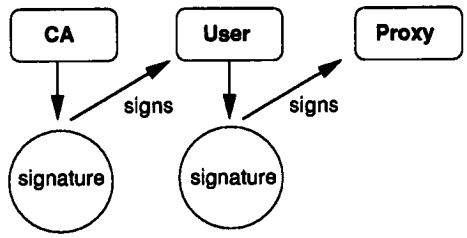

Figure 3.5: Delegation of permissions

Globus also uses proxy certificates which are subsets of the users credentials and can have varying lifetimes and security levels. This allows users to use less strong encryption for basic tasks which reduces the computational overhead. We discuss the performance aspects of varying security levels in Section A.3. The delegation of permissions is demonstrated in Figure 3.5 where the CA signs the Users certificate which can then be used to digitally sign a proxy certificate.

There are several areas including accounting, auditing, restricted access, and data management which Globus acknowledges but does not address in the Version 2.x

3.2 Distributed Systems

Distributed systems research has developed into several distinct fields which each have their own issues and solutions. These include areas such as distributed file systems, distributed databases, distributed resource management, distributed operating systems and distributed real time systems. The Grid is an implementation of a distributed system where emphasis has been placed on data issues and complex security models. This section discusses the concepts behind distributed systems so that we can later show how the Grid has addressed the problems posed [107, 117, 111, 112].

The problems all distributed systems must address can be broken down into six major areas: resource sharing, openness, concurrency, scalability, fault tolerance and transparency [113]. We shall briefly describe each of these and then go into further detail in the subsequent sections.

- Resource sharing describes how multiple users can access resources, such as databases, services and equipment, with minimal disruption to tasks being run by other users and in an efficient manner.
- The openness of a distributed system is the ease with which new resources and users can be added by outside developers and how the system has been implemented e.g. whether it is build on open or proprietary standards.
- Concurrency, the ability to deal with multiple tasks at the same time, can be achieved in many different ways and has been a recurring issue for both hardware and software developers.
- Scalability in distributed systems is based on the addressing schemes used and the potential number of users and nodes supportable.
- Fault tolerance can be separated into three areas: redundancy, recovery, and availability.
  - Redundancy is having nodes in the system capable of taking over tasks if the primary system fails.
  - Recovery is the ability of the system, through either hardware or software, to realise that there is a problem, deal with it and successfully complete the assigned task or notify the user that the task has failed.

- Availability is the ability to discover resources which are capable of performing tasks and ensuring that requests get processed.

- Transparency, which in distributed systems actually refers to the opaqueness of a system, comprises a wide range of aspects including distribution, access, location, concurrency, migration, replication, partition, persistence, failure, performance and scalability.

3.2.1 Resource Sharing

Resource sharing encompasses both locating resources and accessing them. We discuss data location and discovery as a special case in Section 3.3.4, given its relevance to this thesis. There are two groups of methods which allow users and resources to locate hardware and software resources in a system. These are the Push and Pull models.

In the Push model individual services and equipment publish information about their status to one or more sites. This is a highly scalable model as it places the emphasis on the individual nodes registering with a well known site. It has the secondary advantage that it allows resources to be included in the system which have variable reliability without reducing overall reliability. This is known as loose coupling as there is no tight bond between resources.

The Push Model has the disadvantage that security aspects become significant. Which machines are allowed to register with the central site? What level of trust can be placed in the information they are reporting? For example, in a system which does not have an enforced security model to regulate which nodes may register themselves as being available to process information, it would be possible for a malicious user to register their machine and then subtly corrupt results or steal data. There are several potential solutions to this, including sand-boxing and secure identification, which are discussed in relation to specific implementations in later chapters.

In the Pull Model information is requested from nodes by a central system, or its subsidiaries when using a tree model. This has the disadvantage that nodes have to be registered in advance with the central system. This requires an administrative architecture of its own and potentially raises security issues about whether this system can be corrupted.

3.2.2 Openness

The Openness of a distributed system can vary dramatically. At the most proprietary we have systems such as the Microsoft Distribute Component Object Model (DCOM) [53, 54] and .Net [136] architectures which require a specific platform for development and deployment, using tools available from a single supplier with restrictive licencing. In the middle we have systems such as Sun Microsystems Java Remote Method Invocation (RMI) which is platform independent and available using implementations and tools from multiple suppliers but with a single firm controlling the basic standards.

At the most open there are projects such as Mono [83] which is attempting to reverse engineer the Microsoft .Net framework to provide an inter-operable, open source implementation. With the support of developers around the world Mono is trying to allow programmers to use multiple languages and development platforms.

The openness of the Grid has increased over time as there are now several groups developing core middleware for commercial and research goals. Globus was originally developed by a single group, who controlled all development of their implementation and standard. As the user community expanded to include multinational partners and significant investment, the terms of the Globus licence [34] evolved to allow for commercial developments.

3.2.3 Concurrency

The relevance of a systems' ability to deal with multiple discrete tasks simultaneously is proportional to the number of unique tasks it is required to undertake. For distributed systems which are designed to run a single analysis application, such as Seti@Home [71], the ability of the system to deal with dissimilar tasks is less relevant than its security, scalability and performance aspects. For distributed systems such as the Grid the ability of the system to deal with a large number of discrete and dissimilar tasks is a much higher priority.

3.2.4 Scalability

Scalability defines the number of users, nodes, and resources that a system can support. Scalability is an issue which has implications for both software and hardware

as systems experience both logical and physical constraints. If the physicists' requirement, of being able to locate a single byte of data within a Petabyte, is to be fulfilled then this is a significant issue for Grid computing.

Hardware scalability is the ability of the system and its components to address nodes and specific hardware such as memory and disk. Distributed systems have historically dealt with addressing in a variety of ways as there are issues such as, are all nodes aware of the system as a whole, i.e. can every node in the system talk to every other node? If so, can they do it directly? This gives us several possible models: shared memory, message passing and mixed which combines the two.

Software scalability relies on either starting with a namespace which has a significantly large scope and unique aspects, or, having a system which does not require an addressing scheme. Unfortunately, it is often the case that systems which start with the belief that an addressing scheme is unnecessary are often proved wrong. Likewise, systems which think that they are starting with a namespace large enough to fulfil all future requirements are also often proved wrong. The Internet Protocol is a prime example of this as it has had to be changed several times to cope with the increasing number of users and attached devices.

3.2.5 Fault Tolerance

The fault tolerance of a system is its ability to recover from problems in software and hardware without disrupting the user's experience. There are many competing methods of achieving fault tolerance, from Google's well publicised method of using tens of thousands of identical nodes, to the mainframe and HPC communities' use of tested and highly reliable components with multiple redundant systems.

Distributed systems must also consider the systems ability to deal with faults created by users including malicious tampering, such as viruses and worms, accidental faults such as loss of power, damage to equipment, or unforeseen problems such as poorly written code. For example, what should happen if a server fails in the middle of a data transfer? Should the client abort or retry the operation and how frequently should it do this?

3.2.6 Transparency

Transparency in distributed systems covers a wide range of areas which each require significant discussion. These areas include,

- Distribution transparency: Whether or not the distributed nature of the system is hidden at the API layer. This is used to shield developers from concerning themselves with specifying how resources are located or chosen. To do this effectively requires a large initial investment in the infrastructure and the development of APIs and basic services. Reliable and robust services and interfaces have the long term advantage of making it far easier and quicker to develop new applications.

- Selective transparency. When developers can turn off the location abstraction between resources and users. This is vital in situations where physical locations are needed for tasks such as nearest supplier queries. In the early days of the Internet it was claimed that it couldn't deliver pizza. Now, with the ability to identify geographical location by IP address it is possible to customise websites to provide a link to the nearest supplier. Selective transparency can also be useful for performance reasons such as locating data and processing systems which are geographically close. We will discuss this in further detail in Chapter 5.

- Access transparency. This describes whether local and distributed resources are accessed in the same way or must be addressed differently. A good example is printing over IP. This allows computer users to send jobs to their local printer, or one on the other side of the planet, in exactly the same way.

- Location transparency. This deals with the logical view of resources rather than their physical location. For example, distributed file systems typically present their directory structure in the same way as a desktop computer presents its local file system to a user. We discuss how the Storage Resource Broker does this in Section 6.3.1.

- Concurrency transparency. This deals with the ability of users to see what other users are doing. On many Unix based systems users are able to see what processes are being run by other users, and as superusers may have the ability to change the priority of other users' jobs. In a distributed system we may not

want users to be able to access this information but administrators may need this information for performance tuning and to prevent attack.

- Migration transparency is a key issue in distributed systems of the future and a significant problem today. Resources are typically replaced over time as they fail or improved equipment is added, but the effect of changes can have a heavy impact.

- Replication transparency covers both services and data. Replicating data in a system has many performance and reliability benefits, but the consistency of services and data which present a single external interface must be maintained. For example in a system with n copies of a data file, users should either not be allowed to change the data file, or if they do, the file should either present a new external handle or all copies of the file should be atomically changed in the system. There should also be a mechanism in place to ensure that the changes to one copy are propagated amongst all copies in a timely manner.

- Partition transparency. This deals with the parallelism of requests and whether each process in the system is sand-boxed from all others. By making each process discrete we improve the security of the system and reduce the ability to corrupt other processes. This is at the expense of being able to improve work-flow by optimising inter-process communication.

- Persistence transparency. This expounds the concept that a distributed system can use unreliable resources if it is able to hide this unreliability through redundancy. This was one of the original propositions by Paul Baron at RAND in his description of a reliable communications network which could withstand a full scale nuclear attack, [97]. This later influenced the designers of the ARPANET and Internet.

- Failure transparency. What the user and system experiences when a request or function fails. In the same way that we have error handling constructs in programming languages such as *try*, *catch* and *throws* to ensure that if a method fails the program can correct for this, our distributed system needs similar constructs so that it does not enter an infinite loop waiting for data to be returned from a method that has died.

- Performance transparency. The overhead the system has in dealing with requests, facilitating the discovery of and access to resources. The use of a

single resource broker is a potential bottle neck for any system as we will discuss later, but the use of multiple brokers in a system can lead to communications overhead and consistency issues. These factors need to be weighed before implementing a system based on the estimated number of nodes, users and requirements.

- Scalability transparency. Hiding the issues dealing with the speed, size, and geographical scope of the system. This can be achieved by using various algorithms and communications protocols depending on which resources are being accessed to improve performance as one solution does not fit every situation.

3.3 Distributed Data Management

Data management is a significant aspect of Grid computing and an area where the requirements of scientific research diverge dramatically from those of industry. Industry's main concern with regards to distributed systems has until recently focused on the security, financial and processing aspects, and has not taken into consideration the issue of dealing with significant quantities of data. In the following sections we will outline the requirements for distributed storage systems in terms of storage types, distributed file systems, replication, location, security, and data transfer.

3.3.1 Storage Systems

Storage management systems fall into three distinct groups : disk, tape and hierarchical. Individual characteristics make each of these suitable for some tasks and unsuitable for others. However, they are all used in Grid systems, where there are a wide range of demands with competing requirements which must be satisfied.

Disk Systems

Disk based storage systems provide random access to data, typically using rotating platters. These are useful for storing data which is too large for the system's memory to contain, or which must be maintained in the event of a power failure for a non-negligible period of time. However, due to the mechanical nature of these devices, where a platter is spun at speeds between 7,500 and 15,000 revolutions per minute, and the use of a physical head to read and write data, the mean time before

failure (MTBF) is typically 500, 000 hours.

To put this in perspective, the Edinburgh ScotGrid machine has a 24TB disk array comprising roughly three hundred 80GB disks. This gives it an expected failure rate of around 5 disks per year given a 500, 000 MTBF. This is one of the reasons for using a Redundant Array of Inexpensive Disks (RAID) as we are able to maintain a parallel file system which can recreate data from disks which have been lost using various methods but with the cost of reducing the quantity of unique data the system is able to store.

Tape Systems

Tape storage systems provide sequential access to data on a highly durable medium. Tape is an excellent way to store large amounts of infrequently accessed data as it has a low failure rate, large capacity, and relatively low maintenance costs. These costs are typically less than the cost of disk per Gigabyte [63]. This financial saving is at the expense of lower performance for data location (latency) and retrieval (bandwidth) than is achieved by disk. The lower performance is due to the sequential nature of data retrieval from tape which is based on the assumption that both read and write operations will occur in a continuous manner rather than the random retrieval pattern which is possible with disks.

Hierarchical Systems

Hierarchical data storage systems provide a variety of disk and tape systems through a single interface. They have had to face similar problems to the ones the Grid is now faced with and they therefore provide a template for us to base Grid developments upon.

In Figure 3.3 we present an example of a hierarchical storage system. This contains: four front end nodes, which each have their own disk cache; a large, shared disk cache; and a tape based mass storage system. Users send requests to one of the front end nodes, typically using a round robin approach, who will first check if the required data is stored in either its own, or the shared, disk cache. If it is not then the front end node will request that the tape system copy the data to the disk cache. When this is done the front end node copies it to the user. For very large data sets it is possible for the data to be streamed to the users as it is still being read from

releases. Some of these limitations and issues are discussed in future sections.

3.5.2 Information Systems

As we stated at the start of Section 3.5.1 the gatekeeper uses a single default GRAM. Further GRAMs can be made available but this requires users to be able to remember resource names, port numbers and job manager types when submitting jobs. A better, and more productive method of doing this is to use a Resource Broker. These can take generic user requests and allocate them to systems which have the necessary resources if the user has the necessary security privileges. To do this a method is needed for systems to provide information about themselves, and some way of collecting this which can be accessed by both users and Resource Brokers.

The Globus Toolkit comes with both the Grid Index Information Service (GIIS) and the Grid Resource Information Service (GRIS) as part of its Monitoring and Discovery System (MDS). In Version 2 and the early releases of Version 3 MDS was based on the Lightweight Directory Access Protocol (LDAP). Each resource can have a GRIS configured which defines what information about the system is reported, whether this information is only accessible to authenticated users or not, and which GIIS this information is reported to.

The GIIS collates the information from one or more GRISs to provide a single point of information. Each GIIS can be part of a hierarchical system. This allows system administrators to provide information about a cluster, a site and an entire virtual organisation. This allows access to the granularity of information necessary for the task being undertaken.

3.5.3 Data Management

Data management in Globus is based on the Global Access to Secondary Storage (GASS) package. This provides the basic facilities for creating a distributed file system and providing access to data on remote resources using GridFTP. GridFTP is an extension of the FTP protocol which allows for the use of GSI security for authentication, parallel transfers, third-party transfers, partial file transfers and reusable data channels [36, 37, 30].

GASS allows users to create their own data server. This allows them to transfer data and applications to other sites for processing and to transfer data between sites in third-party transfers where the data does not have to pass through the computer the user is working from. Using GASS developers can also build higher level services for replication and retrieval of data.

3.6 Web Services

Web Services have been developed by industry since the late 1990's as a further extension of the object oriented paradigm. They take advantage of the increased use of the Internet by industry, and the business potential for providing basic services in a pay per use model. Web Services address situations where there are a large number of software providers attempting to deal with the demands of a large user base, all of whom want to be able to locate and use various services in a platform and programming language independent way.

Web Services are based on a publish, find, and bind model as shown in Figure 3.6. In this model service providers publish information about their services to one or more service brokers. Users and other services can then request information from the service broker based on the attributes of a service or its reference, and can then bind themselves to an instance of the desired service.

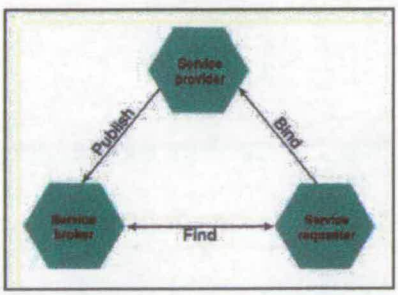

Figure 3.6: The web services publish, find, bind model

This is done using three core technologies: WSDL, SOAP and UDDI.

- WSDL, the Web Services Description Language, provides both a functional and managerial description of a web service. The non-functional information is used by Service brokers to place services within specific service categories

and also contains information about the lifetime of a service. The functional aspects of the WSDL document are used to tell Service requesters the format that requests must take, the format results will be returned in and the options for negotiating protocols and operational aspects.

- SOAP, the Simple Object Access Protocol [74], provides an XML based method of passing messages in distributed environments. These messages contain information about their internal structure, i.e. its encoding rules, and can include remote method invocations as well as information.

- UDDI, Universal Description Discovery & Integration, is used to catalogue Web Service specifications and to match these services to Service requests. UDDI is a resource broker which matches Service requests to matching resources. As such there are three models for its implementation [81] :

    1. The Promiscuous broker. The broker will allow anyone to register Services with it and allow anyone to request them. This does not guarantee that the Service is what it says it is, nor that users are who they say they are.

    2. The Authenticated broker. The broker authenticates that both the user and the Service are who they claim to be. This level of security ensures an audit trail which can show who requested a Service but does not address many of the security issues faced by both research and industry.

    3. The Fully Authorised broker. With fully authorised brokers we arrive at a scenario which would allow fine grain mapping between users and Services based on their role and level of authorisation. Both users and Services would have to prove their identity and the level of accounting and auditing in the system would allow for meaningful financial accounting for resource usage.

3.7 Grid Services

Grid Services were developed in response to industry's development of Web Services to ensure that the academic and research communities could make use of the support and facilities industry were developing while still tailoring these services to their specific needs. The Grid Service framework was first announced by Ian Foster at the Global Grid Forum meeting in Toronto, Canada in February 2002. At a plenary

session he introduced a paper, the "Physiology of the Grid" [100], which extended the original work done by the web services community.

In this paper they defined an Open Grid Services Architecture (OGSA) which would allow inter-operation between Grids and Web Service frameworks such as .Net, SunONE and IBM WebSphere. The requirements for a Grid Service are based on six basic web service portTypes or Services: GridService, NotificationSource, NotificationSink, Registry, Factory, and HandleMap. Further higher level services could be built upon these to provide necessary functionality. We will present a brief description of each of these core Services and then discuss them in the context of the Globus 3.x toolkit which has provided a reference implementation.

- GridService. The GridService portType provides facilities to retrieve and query data about a Service, set the termination policy for a Service and destroy a Service. This allows higher level Services to match their needs with the Services available and to ensure that a Service will be available for the duration of their requirements.

- NotificationSource allows Services to request notification about events which can be used to orchestrate other Services. This could, for example, be used as part of a resource reservation system which supplies notification when a resource becomes available.

- NotificationSink. When Services have signed up to a NotificationSource to receive information about a Service a method is necessary to provide this. The NotificationSink provides the asynchronous messages required to keep other Services updated about the local ones.

- Registry. The Registry provides facilities for soft-state registering and deregistering of Services. This Registry acts in a similar manner to UDDI in Web Services in providing a well known location for user Services to locate Services based on their capabilities.

- Factory. The Factory acts as a persistent web service which will spawn Services on request. This maintains a *state* which is typically not available in standard web services but is necessary in Grid environments.

- HandleMap. The HandleMap provides a mapping between a Service type and the instance(s) of it. This is similar to the logical to physical file name

mappings used by Replica Location Service for LCG.

3.7.1 Globus 3.x

The third version of the Globus Toolkit (GT3) was released in June 2002 [33, 139] implementing the specifications laid out in [142]. In this version of the toolkit the Grid Resource Allocation Manager (GRAM), which was used in Version 2.x of the toolkit, is replaced by the OGSA Registry and Factory. This still provides the mapping between requests and a local account, but extends the ability of the system to deal with auditing and accounting functionality.

Information reporting and presentation in GT3 is still managed by MDS, however the new version of MDS is based on using relational databases to store information rather than LDAP. This extends the system's ability to mine data for information through the use of further services which are less static than the LDAP model.

GridFTP is still supported for data transfer and the Reliable File Transfer (RTF) package has been added. RTF utilises GridFTP to provide a guaranteed data transfer. The Globus Replica Location Service (RLS) has been included for groups which have not developed a stand-alone system of their own.

GT3 has been widely used in the UK e-Science programme outside the physics community. Given the existing timelines it is unlikely that GT3 will be widely adopted by physics.

3.7.2 WS Resource Framework

The WS-Resource Framework (WS-RF) specification was announced in January 2004 as a further extension to work on Grid Services which would bring it closer to the Web Services standards. This framework outlines six further web service specifications. These are: WS-ResourceLifetime; WS-ResourceProperties; WS-Notification; WS-RenewableReferences; WS-ServiceGroup; and WS-BaseFaults. It was also announced that these would be implemented in the Globus Toolkit 4.0 which is scheduled for release in January 2005. Further information about these are available from [82, 27].

3.8 Conclusions

Access to processing capabilities is one of the first issues that Web Services and the Grid have addressed but with significantly different aims. While Web Services started dealing with transaction biased processing, the Grid has from the outset been more concerned with access to significant processing capabilities. We believe that the Grid can learn from the experiences of Web Services in providing lightweight authentication, as we discuss in Appendix A. Likewise, Web Services can learn from the experiences of the Grid in terms of authentication systems and delegation which we discussed in Section 3.5.1.

Networking is approached in different ways by all three groups. Grid developers expect dedicated multi-Gigabit connections between sites. P2P developers develop their own logical networks, and Web Services assume generic, limited bandwidth connections. In Section 5.2.4 we present our work on benchmarking the network capabilities of the Edinburgh ScotGrid site and how the performance is affected when we move from dedicated systems to regional and international ones.

Data Management is currently more of an issue for P2P systems and the Grid. P2P systems typically assume a different usage scenario for both the users and applications than Grids do. The initial expectation for the Grid was to have a small number of very large files. From the experience the Grid community has gained from deploying systems this is not necessarily the case. There are lessons which can be learnt from P2P systems for data discovery, provenance files and dealing with data corruption.

Information systems are an issue for the Grid, Web Services and P2P systems. Each of these groups have different needs but a common requirement is for more work to be done in this area. This is outside the scope of this thesis but a worthy area of future research.

We conclude by saying that Grid Services are likely to have a significant impact on the LHC related particle physics experiments in the near future, and may have an effect on existing production experiments. The European DataGrid and the LHC Computing Grid have sites that have started to specialise in some functions, while others are provided using standard software. It makes sense to move services

such as VO membership lists and resource information to a Grid Service Model, while functionality such as database access can be replicated at regional sites. With existing timelines GT4 is a likely candidate for future development work.

Chapter 4

Data Issues for the BaBar Experiment

BaBar is an active experiment based at the Stanford Linear Accelerator Center (SLAC). It is studying the millions of B mesons produced by the PEP-II storage ring. The BaBar Collaboration consists of approximately 600 physicists and engineers and involves 80 institutions in 10 countries. As BaBar is an active experiment, its developers face many challenges when enhancing the data management facilities as they must also support production physics research in parallel.

This chapter presents an overview of the development of the BaBar computing model and work we have done on the analysis of the existing BaBar data distribution software. We use this to consider how existing software can be adapted for use in a Grid environment. The BdbServer++ project is presented as an early feasibility study of migrating existing software to the Grid. As part of this we provide the results from testing the performance of the Globus Toolkit in Appendix A.

4.1 Motivation

Our motivation in developing the BdbServer++ project (Section 4.5) and the BBGUtils sub-project (Section 4.5.1) is to gain experience with the computing model of an active particle physics experiment. This can then be applied to the development of future experiments, specifically LHCb. The aim is to discover what issues face developers in moving existing applications to a Grid framework; and to investigate and develop methods of retrieving data using the Grid. To do this however, we

Figure 4.1: A timeline of the BdbServer++ project

must first present a context for our work which is best defined as an extension of the continuing evolution of the BaBar computing model. This is assisted by Figure 4.1, which presents a timeline of our work in the context of other developments.

4.2 Overview of BaBar Computing

Whichever computing model BaBar uses there are six computing tasks the system must be capable of supporting:

- online event processing
- data reconstruction including
  - prompt calibration
  - event reconstruction
- simulation production
- data storage
- data skimming
- data analysis

Theoretically none of these tasks require the computing resources to be physically located near the detector. However, due to the limitations of current hardware and networks, online event processing has been traditionally performed at SLAC as it allows rapid feedback to the detector experts which is vital for the experiment and reduces the risks associated with network availability. A brief outline of each of these is presented in the following descriptions.

Online event processing (OEP) [8] is a subsystem of the BABAR online system. Its responsibilities begin when it receives complete events from the Data Flow system, specifically the Event Builder component. OEP encompasses the operation of the Level 3 trigger algorithm, event data quality assurance, and other actions of the online system dealing with events, short of their complete reconstruction.

Data reconstruction includes prompt calibration and event reconstruction. Prompt calibration is a one pass event reconstruction that only reconstructs special calibration event types, such as e+e- pairs, muon pairs, etc. The more extensive second

pass event reconstruction, which reconstructs all events using the calibration constants from prompt reconstruction [65], is done at INFN Padova. Data is transfered to Padova, reconstructed and shipped back to SLAC in under 24 hours.

Simulation production (SP) [72] is a highly, or *embarrassingly parallel*, task where nodes are independent of each other and do not require interprocess communication. Simulation production is used to create an accurate representation of events occurring at the interaction point in the detector. Using the GEANT4 package [28] to model the detector SP reproduces how these events would propagate through the detector and the response of the detector to final state particles. Events which would satisfy trigger requirements and would be recorded by the detector, are stored and can be analysed as if they were real data. The simulation production teams typically generate billions of events every year. In 2002 SP4 generated 1.5 billion events, in 2003 SP5 generated 2.2 billion events and currently[1] SP6 has generated 1.7 billion events [7].

Data storage [6] is concerned with the reliable and robust transfer of data between sites, without data corruption occurring. This can be done using many protocols, hardware media and architectures. As a worst case scenario data can be sent by tape between sites. Data distribution issues facing the BaBar experiment are discussed in Section 4.4.

Data skimming [73] is used to provide pre-selections which contain collections of pointers to data which is logically grouped. These skims allow subsets of data to be retrieved by users for analysis. By retrieving only the data that they require, rather than larger or complete sets, which the users must then filter, skims reduce the amount of data extracted from the database and the amount transfered over the network.

Data analysis [5, 39, 68] is performed by physicists primarily to produce physics result from acquired data. Using simulated data for data analysis allows physicists to ensure that the simulation models used give an accurate representation. SLAC, RAL, and IN2P3 provide analysis farms for the collaboration which all members are entitled to access. Institutes also have their own resources which allow authorised researchers to analyse data without having to wait for the analysis farms at the

[1] November 2004

major sites.

4.2.1 Tiered Computing for BaBar

In each BaBar computing model there is a standard nomenclature used to describe the capabilities and intentions of the various sites. This is based on a three tier model with the sites classed as being either Tier A, Tier B, or Tier C.

A Tier A site is a national or regional centre with significant computing and data storage capabilities. These sites are expected to provide accounts for all BaBar users and provide at least a minimum amount of data storage and computing capabilities at an agreed level of performance and reliability. Currently there are 5 Tier A sites housed at SLAC, IN2P3 (Lyon), INFN (Padova, and soon to include Bologna), RAL and Karlsruhe. To provide an example, the RAL site provides 700 Intel CPUs (450MHz-2.6GHz) running Linux, 80TB of available disk space and 60TB of Tape [79, 66].

A Tier B site was the designation given to a centre which provided resources to groups outside their own administrative domain in the CM1 model. Tier B sites were regarded as candidate Tier A sites. They had a reduced requirement for the quantity of resources they were required to make available and lower expectations of guaranteed uptime and access. CASPUR and RAL were the only two official Tier B sites when BaBar started taking data in 1999. RAL later expanded to become a Tier A site and CASPUR became a Tier C site when INFN became the official Italian Tier A centre. There are therefore, no Tier B sites in the CM2 model.

A Tier C site is an institute or research group which uses the data generated by the BaBar experiment for physics analysis. These sites have resources to analyse data and also, in aggregate, provide most of the simulation production for the experiment. There is no requirement for Tier C sites to provide accounts to users outside their administrative domain nor are there agreed levels of service they must support.

4.3 Evolution of the BaBar Computing Model

The computing model used by BaBar has evolved over the life of the experiment as the membership of the experiment, the resources available, and the technologies

have changed. The initial BaBar computing model, later called CM0, was used in the run up to the start of the detector actively take data (1996 to 1999) and during 1999 and 2000 in production. This model used the Objectivity [60, 84] ODBMS for all the experiment's data storage requirements, and used a centralised model for data processing and simulation.

With the start of active data taking, deficiencies were found in the way the experiment was storing its data, based on how users were accessing it. The complexity of the initial model resulted in the second computing model, CM1, being developed to allow BaBar to move from using an Objectivity database to a ROOT-based event store for data analysis. CM1, which was in production use between 2000 and 2003, maintained Objectivity at the larger computing centres but with the affiliated institutes and some large centres using the KANGA data format [45, 46]. Analysis and simulation became more distributed but data processing was still performed centrally.

In the third computing model, CM2 (2003-present), ROOT [69] is used for the majority of data storage at both large centres and institutes. ROOT has been used to expand the EventStore to include the Mini data format and Objectivity is now used only for the online and conditions databases.

Currently there are discussions as to whether to move to a Grid based computing model. This may require a further change in the data storage method used.

4.3.1 The Original Model (CM0)

The original BaBar computing model was developed during the mid and late 1990's and was actively used for data challenges in the run up to the start of BaBar collecting data, and for the 1999 and 2000 production runs. This model was based on a SLAC-centric design where all the computing tasks, except final data analysis, were performed at SLAC. The Internet was used for transferring highly reduced, "micro", data sets to large centres in Europe after they had been added to the Objectivity database and skimmed on the computing farms at SLAC.

Analysis farms at IN2P3 and at the Tier B sites, RAL and CASPUR, imported data from SLAC to their own Objectivity databases. These sites allowed collaborators

access to them so as to perform analysis on the data stored there. Institutions could also extract data from the Objectivity database at the Tier A and B sites and import it to their local cluster for analysis.

The key feature of this model was the use of Objectivity for the EventStore. When the system went into production use it became obvious that there were several issues with this. These included licencing costs for individual institutions, the difficulties of developing and supporting Objectivity databases at each site, and Objectivity's lock management when supporting multiple users. Of these the lock management was the biggest problem.

The Objectivity system uses software locks to prevent users from overwriting data while it is in use and from reading data which is being updated. The system had been designed for significantly different usage patterns to those of particle physics. As a result, the lock servers were unable to keep up with the rate at which users were creating and releasing locks and many locks were becoming separated from their users requiring system administrator intervention. This forced those involved with developing the data management and analysis capabilities of BaBar to start looking at alternatives.

4.3.2 A transitional model (CM1)

When the first computing model went into production use the centralised system for data storage and management was unpopular with the users. Resources at SLAC were being fully utilised resulting in researchers having to wait for access. With computing capabilities increasing relative to price, and the Linux Operating system gaining popularity, institutions were starting to acquire significantly more local processing and data storage and were reaching the point where they controlled more resources than the Tier A sites.

The Tier B centres were either disbanded or extended to provide Tier A facilities leading to a situation where there is now a Tier A and Tier C but no Tier B as we show in Figure 4.2.

The working group which analysed the computing model concluded that the main problem with CM0 was the method by which data was stored. The report concluded

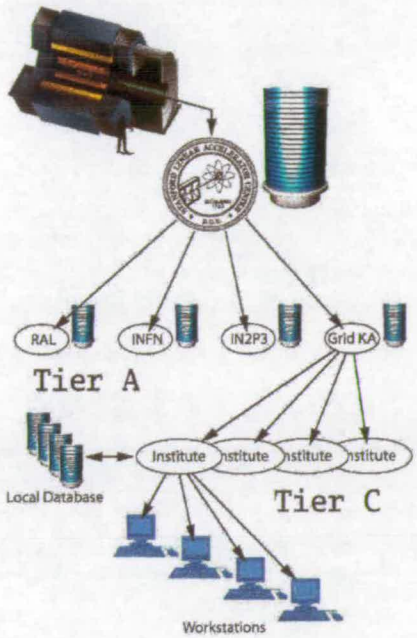

Figure 4.2: Current organisation of BaBar computing resources

that the Objectivity database is inappropriate for remote sites as it is difficult for users to install and configure. Additionally, the number of licences needed for every institute to make use of it at Tier C, and some Tier A, sites would be prohibitively expensive. The decision was made to migrate data storage at remote sites to the Kind ANd Gentle Analysis (KANGA) format [17].

4.3.3 The Current Model (CM2)

The purpose of the move to the second Computing Model was four-fold: to create a new EventStore to replace Objectivity at all sites; to create a new analysis framework to replace the one based on Objectivity; to create new data content at remote sites for analysis (Micro and Mini); and to improve book keeping and task management. Development toward these objectives has been an ongoing task and is partly assisted by outside developments, such as the Grid, where generic functionality is provided.

The current model, CM2 (2003-present), has minimised the use of the Objectivity database to certain Tier A sites where the online and conditions databases are maintained and to Tier C sites which are involved with simulation production where the conditions database is required. ROOT has replaced Objectivity as the event store

for experimental data from the detector, and for simulation output. This migration has also enabled the creation of the Mini format which is a condensed description of detector objects. The migration of the data format has allowed a move from the batch processing of jobs to interactive analysis tools which allow rapid feed back to physicists. This allows researchers to walk through their code to discover bugs before valuable CPU time on the batch farm is used.

Data analysis in CM2 is now fully distributed. This allows more resources to be utilised than would have been possible under the centralised model as institutes resources can be used for a limited time before being returned to their core tasks. Simulation production, which has linear scalability, is now primarily conducted at the institute level which allows Tier A centres to concentrate their resources on reconstruction, skimming and calibration.

4.3.4 Time for a new Grid computing model?

With the large amount of development work which is going into the Grid to support the LHC experiments there is value in BaBar investigating if it is possible to migrate to the same architecture. However, as has been repeatedly stated, BaBar is an active experiment where users want immediate results, and are intolerant of system outages and failures.

The LHC experiments can gain from the involvement of BaBar, as an active experiment which has a significant quantity of data and computing requirements similar to those anticipated for LHC. In terms of testing the capabilities of the systems being developed, LHC developers would also gain early experience of the types of complaints and requests they are likely to receive when the Grid becomes active with LHC-related tasks. In the next section we present our work as part of the effort to discover if this is achievable, and what lessons can be learnt for the LHC experiments.

4.4 Data Distribution

The distribution of BaBar data has three distinct areas: transfer between the Tier A sites; transfer from Tier A sites to Tier C sites; and transfer from Tier C to Tier A sites. We discuss each of these in terms of what the experiment is attempting to

achieve and what areas have potential for enhancement by using the Grid.

4.4.1 Tier A to Tier A transfer

Data is transfered between Tier A sites for a variety of reasons including:

- Off-site storage
- Data replication
- Performance
- Site specialisation

Off-site storage is widely used by both academia and industry to ensure that a major disaster can be recovered from. Typically it is required for events which have a small possibility of occurring but have done so in the past such as flood, fire, electrical damage, human error, earthquake and equipment failure. These events have been seen at physics laboratories in the past and are likely to occur again.

Replication of the data has several advantages. We can reduce costs by having a copy of data on each side of the Atlantic. This reduces the amount of data transfered over what is still the most expensive segment of the network. Performance is also improved by reducing the network latency of requests, and scalability is enhanced, allowing for more analysis sites to be supported.

Sites can specialise in terms of function, i.e. simulation, event reconstruction, skimming, or data storage. This allows them to build expertise which can be applied to future experiments, and to improve productivity for the existing one. The most obvious example of this is data reconstruction at Padova. Data from the BaBar detector is initially processed at SLAC and approximately 600 Gigabytes per day is transfered across the Atlantic for data reconstruction in Italy. The results, which constitute approximately 100 Gigabytes of data per day, are then transfered back to SLAC. This process now takes under 24 hours for both of the data transfers and the processing time at Padova.

4.4.2 Tier A to Tier C transfer

In Tier A to Tier C data transfers users request that the data they are interested in be copied to a site to which they have access, and which has the capability to

analyse it. This type of data transfer raises the problem of whether to move data to the processor or the processor to the data, i.e. do we copy the data to Tier C sites for analysis, or supply more computing resources at the Tier A sites? There are many ways to solve this, and any solution must take the characteristics of both particle physics and the specific experiment into account when addressing it. This makes it an area with many opportunities for enhancement and optimisation which we shall discuss in Section 4.5 as part of the BdbServer++ project.

4.4.3 Tier C to Tier A transfer

The majority of data transfered from Tier C sites to a Tier A centre is to register simulation output for data storage. During simulation production runs Tier C sites typically transfer a regular block of data in the hundreds of Gigabyte range. It would be possible to transfer smaller blocks of data continuously, but the potential overhead in terms of CPU time for creating and closing a large number of connections may negate the increased use of the network connection, and adversely effect users at SLAC who are attempting to connect to remote sites. This area can also benefit from using the Grid and is partially addressed in Section 4.5.1.

4.5 BdbServer++

The original BdbServer software was developed and used in the CM1 computing model [10] as part of a suite of tools for data management. BdbServer allowed users to request copies of data held in the BaBar database (Bdb) by submitting requests via email. These messages were sent to a mailbox at SLAC which was scanned by a simple *cronjob* script every fifteen minutes. This script was responsible for the actual executing of the request, notifying the user when the data had been retrieved, and reporting its temporary location. Users could then copy this data to another machine at SLAC or to a remote site for analysis.

In Figure 4.3 we show an abstract of the steps taken to locate, retrieve and analyse data in the Objectivity model. All of these steps can be conducted by the user directly, however the role of BdbServer was to automate the more time consuming parts, such as the actual data extraction from the database.

Following the flow of the diagram, users first identify, or Locate, the collections or

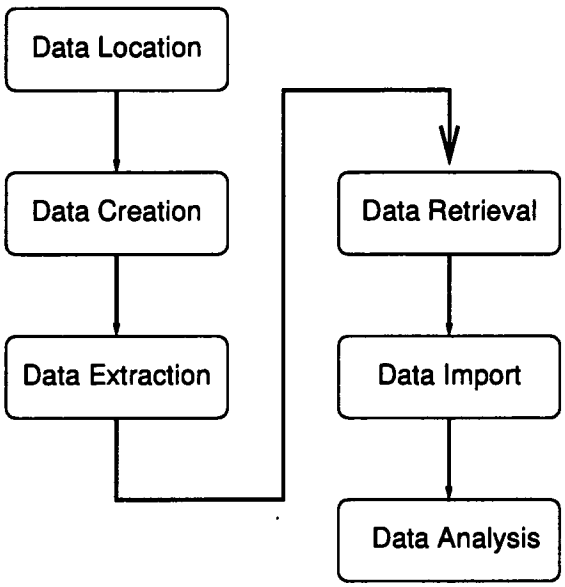

Figure 4.3: Workflow for BaBar analysis

data types which contain the data they require by using either the colldb, skimData, or BdbInspector packages. Each of these is aimed at differing levels of data granularity, i.e. the degree of detail or precision contained in the data. Colldb uses the Objectivity database to present collections of objects to the user. skimData uses the BRORA (BaBaR ORAcle) database to find data files based on their attributes. BdbInspector allows the user to navigate through the Objectivity database and is typically used as an API for applications.

Having located the initial data, users can then Create their individual collection by emailing their request to BdbServer. In the email they must specify the federation they require data from e.g. simulation or analysis, or the collection and type of data they require, such as Tag, Micro, Mini, Rec or Raw. In addition, data from the initial collections can be copied into a new collection using *BdbCopyJob*. This can take a significant amount of time, depending on the size of the request and the load the database server is under.

The new collection will also be stored in the database, so users must request that their data is Extracted from the database and copied to disk. The method of Retrieving this data to the chosen analysis site is left to the user to organise. Once copied or moved over the network, the data must be Imported and registered into the local Objectivity database before an analysis can be run.

This system is obviously labour intensive. Even with BdbServer managing the data creation and extraction phases, the user is still expected to organise the location, retrieval and import. Most of these stages are prone to user error, especially with new and inexperienced researchers. BdbServer was not considered to be an effective long term solution. The decision was made to investigate whether the Grid would be a better system for data retrieval. As part of this the BdbServer++ project was initiated.

4.5.1 BdbServer++

The BdbServer++ project initially defined seven questions [120] which mainly dealt with data extraction from Tier A sites, but also included the location of data prior to extraction, and its transfer after extraction (Figure 4.4). These questions needed to be answered to help us decide whether the Grid was a viable model for BaBar. They were:

1. How should we deal with queries about the creation or extraction of a collection? (*location and extraction*)
2. How should we deal with duplicate queries from different users? (*location and extraction*)
3. How should we deal with requests to cancel a creation or extraction request? (*extraction*)
4. How should we deal with requests to copy large fractions of the dataset selected with very loose criteria? (*extraction*)
5. Should we be able to deal with priority requests? (*location, extraction and transfer*)
6. How should we manage the temporary staging area for BdbServer and should there be lifetimes in the staging area? (*extraction and transfer*)
7. How do we deal with a large number of requests? (*extraction and transfer*)

We looked at two different methods of achieving these aims: a web interface and a command line system. The development and issues associated with each approach are discussed in the following sections.

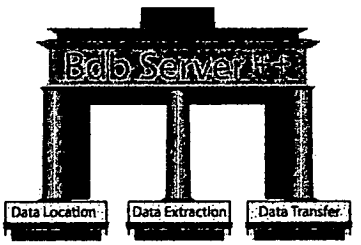

Figure 4.4: The Scope of BdbServer++

Web Interface

At the start of the BdbServer++ project there was significant interest in using web interfaces to provide users with a simplified view of complex functionality. Other groups were working to providing web interfaces for simulation production and data location, and this seemed like a logical route for BdbServer++. Because of this, the initial solution proposed for BdbServer++ was to create a web interface. This would interact with various services, including an authentication and authorisation system, and a job submission system as shown in Figure 4.5.

We assumed that BdbServer++ would eventually re-use the authentication system from the BaBar collaboration website, but would initially provide its own authentication for a small group of test users. After the initial problems were located and addressed we could then migrate this part of the system to the experiment's authentication database. In effect this would mean that users would not have to register specifically for BdbServer++, and would not have to remember an additional password. This also gave us the ability to disable the system in case serious problems were found with it while in development.

We envisaged that data would be extracted from the database using the existing job submission system, specifically the batch queue. This mimicked the way BdbServer had submitted requests, and was therefore well understood and known to work.

After discussions with users and other developers, we decided a useful property for the system would be to have a record of user's details including name, email address and preferences. This would save the user time when making a request, as commonly used information, such as the email address needed to report the final condition of a request, could be stored rather than requiring its re-entry with each request. If we stored these details, the infrastructure required would also give us the option of

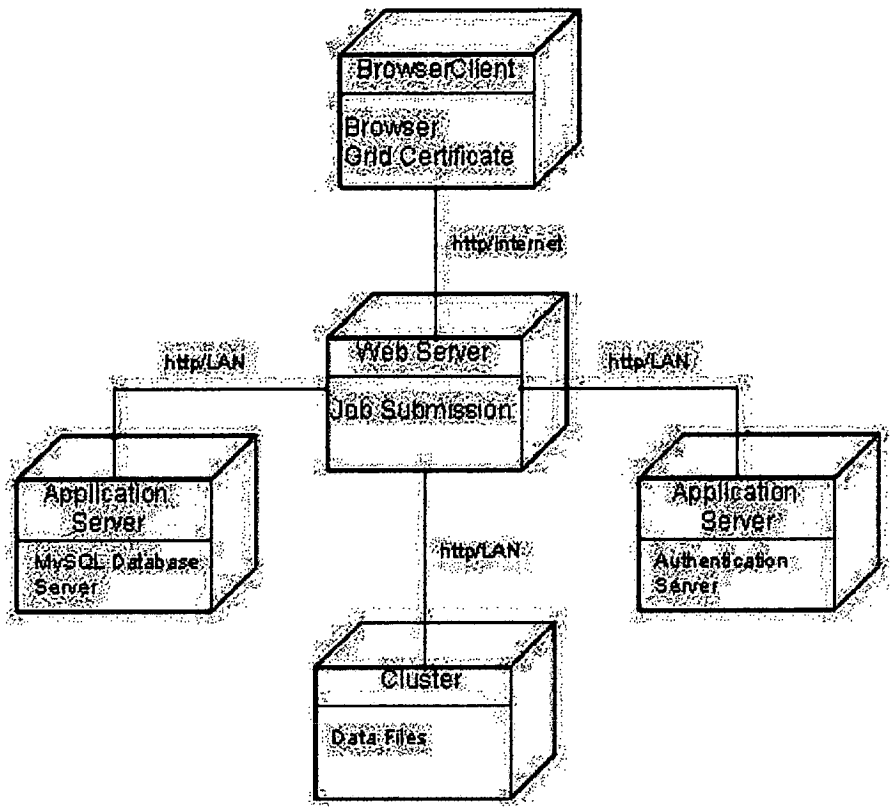

Figure 4.5: A Deployment Diagram of BdbServer++

providing the user with the ability to store prepared commands for regular tasks.

We initially considered using a file to store this data, but reached the conclusion that the information would become too large to be human readable. We also required the data to be queried in a random manner which suggested a database would be a better method. Using a database would also provide us with the ability to create an audit trail of every request made.

It was generally felt that early Grid software failed to provide adequate audit and accounting functionality for many organisations. At the start of this project attempts to improve efficiency through data replication and caching were based on the administrator's intuition and experience, rather than empirical evidence. Giving BdbServer++ an audit trail was intended to provide this evidence. The audit trail would also allow administrators to see what each user had been attempting to

do, to aid in the resolution of problems.

To manage the information required to do this, we decided that a database was necessary. Initially we created a new MySQL database within the existing system at the University of Edinburgh. This had an administrator and user account associated with it. The administrator account was intended to be used for direct access, to allow the deletion and modification of existing data. The user account was intended to be used as part of the web interface. This account had much more restricted functionality, and was initially restricted to read only access to the database.

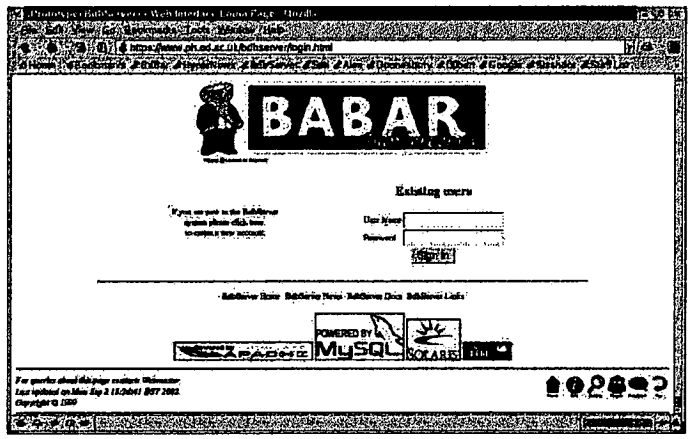

Figure 4.6: The BdbServer++ Web Interface

Figure 4.6 shows the initial screen of the web interface we developed. It required users to log into the system in a manner similar to that used by many websites. In doing this the user created a *cookie* which allowed the system to match the users session with their details in the database. A further advantage of using a database was that it was possible to extend the range of options open to the user, without having to re-write web pages or releasing new software to multiple sites.

Several issues were discovered with this model which led to it being abandoned. We misjudged the target audience in terms of their technical competence, and the intended use for the software. BaBar users wanted any system developed to be compatible with their existing scripts and software. The use of a graphical interface, either web-based or as a Java applet, would have resulted in the need for a command line version or API to be developed in addition to the web interface.

A more serious problem with the proposed system was that users were asked to upload a copy of their proxy certificate to the server. This would allow the server to masquerade as the user and submit jobs to gatekeeper systems at the appropriate Tier A site. The security implications of this were significant, and in early 2003 no short term solution could be found which would be suitable for the web interface we had developed. As of late 2004 there are several solutions to this problem which we look at in the conclusions.

Command Line Interface

A command line interface was seen as being a reasonable solution to the many problems the web interface had raised. It would provide the required functionality of being able to retrieve data while having the ability to be included in other scripts and programs. In Figure 4.7 we demonstrate how we envisaged the flow of data and instructions using this software.

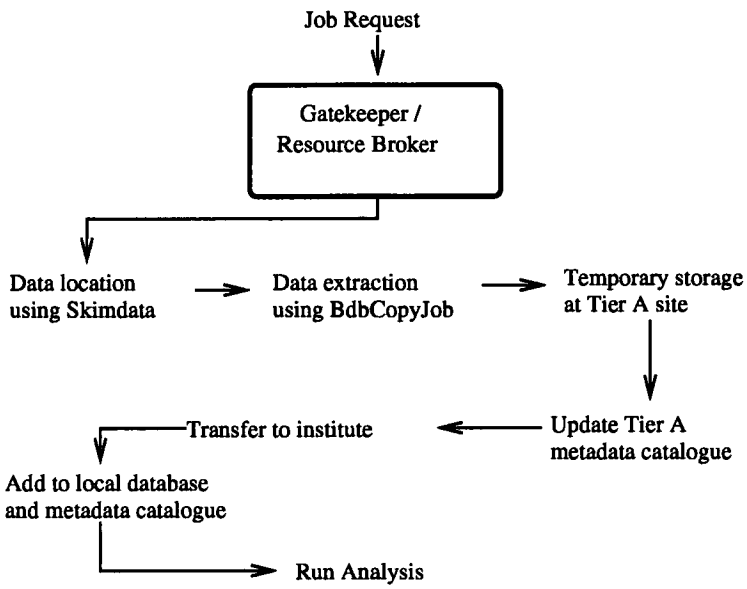

Figure 4.7: Workflow for BaBar analysis using BdbServer++

We started by configuring a personal Globus gatekeeper, based on the 2.1.0 release, using a desktop machine at SLAC. The developer who started the gatekeeper could then submit jobs which were run using their local user account. This avoided the need for host certificates for the machine, as the gatekeeper could be run off a personal certificate, and for root access, as the software could be installed in the user's

space. This was intended as a development system, and a dedicated system would be used in production with the necessary host and service certificates and system installation.

We ran each command by submitting them sequentially to the gatekeeper. After we had proved that this method worked we condensed them into a single command file based on the Globus Resource Specification Language (RSL). This provided the basis for us to create a Job Description Language (JDL) file which could be submitted to an EDG Resource Broker for execution. These tests proved that it is possible to run the existing software using the Grid as a submission mechanism.

There were several remaining issues related to the use of user accounts, and the batch processing nature of particle physics computing. User accounts can be created for individual use or for shared usage. Experience has shown that shared accounts make it difficult for system administrators to prove who is responsible for the consequences of commands which have been run. The Globus software solves this by mapping users to their personal account.

This has been an acceptable method at Tier A sites, where all users have the right to an account, but Tier C institutions are not prepared to deal with the bureaucracy and potential legal entanglements of giving all researchers in a collaboration individual accounts. The BdbServer++ project made use of the fact that requests were run from a stated user's account. The problems posed by using generic accounts increased the complexity of the required solution.

The batch nature of particle physics computing also raises issues. It is not always possible to specify at the start of a job where the output will reside at the end, as system resources change over time. Likewise issues such as advanced reservation of processing, network capacity and data storage are only now, two years after the end of the BdbServer++ project, being addressed.

Some of the issues raised by the feedback from the people who tested the system were the problems of installing, configuring and running the Globus client software, and the performance of the Grid. As part of this we developed the BaBar Grid Utilities package, BBGUtils, which is discussed later. We spent some time looking at the performance of the Globus gatekeeper, in terms of its performance under

various security regimes and with varying load. This is presented in Appendix A.

BBGUtils

The BBGUtils package was developed to assist users and system administrators wishing to use the Grid for job submission and basic data transfer. At the time BBGUtils was developed, both groups were unsure of the security risks associated with opening ports in their firewall to allow job submission and resource monitoring over the Grid, by remote users submitting requests to their machines. After consultations with potential users a requirements list was created :

1. The software should be installed using a non-privileged account so that students and researchers without *root* access can install the software without requiring system administrators assistance.

2. It should be possible to install from source, as users expressed interest in running a variety of Unix derived operating systems.

3. Installation from binaries should be included as an option, primarily for Red Hat Linux and Sun Microsystem Solaris as these were the most widely used operating systems.

4. It should not open ports permanently on the user's machine.

5. It should support the certificate authorities associated with BaBar.

6. It should be well documented

7. It should have minimal complexity and a low learning curve to install and use

The BBGUtils package was developed using a Perl script which ran from the command line. It re-bundled the three client packages of the Globus 2.3.2 release[2], the Grid Packaging Toolkit (GPT) 2.2.5, and the Certificate Authority information for the BaBar collaboration as a single entity. With this, users had the ability to install the software in their home directory. They were also able to install the software in a system directory if they had *root* privileges, and wanted multiple machines or users to share the core software over NFS.

[2]This was the most recent stable version at the time of release

The software gave the user the option of installing from either binary or source, and the script was capable of distinguishing between Linux and Solaris. The CA certificates for the groups associated with BaBar were installed in a non-privileged directory (*GLOBUS_LOCATION/share/certificates*). Adequate user documentation was packaged with BBGUtils and was available on the Web [9]. This software package improved relations with the system administrator community who were wary of this new and relatively untested software. The package was used at several institutes in the US to provide Grid facilities for researchers.

Issues such as user certificate revocation lists, and updates to the virtual organisation membership list, which are needed for gatekeeper systems, were unnecessary for BBGUtils, as the software allowed users to submit jobs to other sites, but not to a permanent gatekeeper on their own machine. CA certificates typically have a longer lifetime than user certificates (5+ years vs. 1 year), and it was assumed that the users would have to upgrade their software before this issue needed addressing.

4.6 Summary of BdbServer++

The BdbServer++ project answered the questions which were initially defined, and created new areas of investigation which are discussed in the conclusions.

> *How should we deal with queries about the creation or extraction of a collection when they are in progress?*

The existing BdbServer software did not offer this facility and it was decided that this functionality was not a priority. As requests are submitted to a batch queue users are able to query what the status of their request is in terms of queued, active, complete or failed. These four answers are sufficient for almost all of the average users requirements.

> *How should we deal with duplicate queries from different users?*

In its final command line form BdbServer++ provided an interface to existing software which does not have this functionality. The command line version of BdbServer++ does not maintain historical data about the users requests. To do this would have required rewriting the Globus GRAM system, which would have then either needed to be accepted by Globus for inclusion in future releases, or it would

have required us to continually modify BdbServer++ as new Globus versions were released.

Another concern is the problem of object databases defining what a duplicate query is. It is relatively simple to discover if one file is the same as another or if several users are requesting the same file. However, a database is far more complex. Data which has been extracted already could be a subset of the data required, or data requested by various users could be combined to create the required set. This is an interesting research area, but would have taken too long to be useful to this project

How should we deal with requests to cancel a creation or extraction request?

As requests were submitted to batch queues, standard commands which could be invoked from the local system or through a Grid interface, are available to both the users and system administrators.

How should we deal with requests to copy large fractions of the dataset selected with very loose criteria?

No clear and easy way to do this was found. The disadvantages of capping users ability to request data were seen to outweigh the advantages of having a simple user interface. A system to cap request size would also be difficult to implement with an object database.

Limiting the size of data retrieval would have required further software on the server, either as a separate package or as an extension to the existing software. Both of these options could present a potential area for bugs to be introduced, a risk which should be minimised in a production experiment.

Should we be able to deal with priority requests?

The project concluded that priority requests were unnecessary. There are several points in the existing computing infrastructure where system administrators can cancel requests or increase their priority. The ability to have multiple queues was discussed, but by this stage the development of the EDG Resource Broker was at a point where it was decided that the Resource Broker layer would be a better location for optimising requests.

How should we manage the temporary staging area for BdbServer and should there be lifetimes in the staging area?

The BdbServer++ project concluded that users should not be expected to delete their temporary files themselves. The system in place with BdbServer deleted files when it was told to by an administrator. From this starting point BdbServer++ concluded that there were four options for dealing with the management of temporary files.

1. Have administrators monitor available disk space and delete files when necessary.
2. Run an automated script on the server to monitor available disk space and delete the oldest files when space was needed.
3. Run an automated script on the server which would delete files older than a set time, e.g. one day or week.
4. Adapt the data transfer package to delete files after copying them to a remote location.

Of these the first solution was adopted, as system administrators have a greater level of intelligence than scripts, and are more able to deal with changing situations and needs. The fourth solution, where the package moves the file rather than copying it, was considered, but at the time the software would not allow us to force this. The question of temporary files is still an area of active research, which is discussed in Chapter 6 in terms of the Storage Resource Manager specification, and the implementations which are currently being developed.

How do we deal with a large number of requests?

As we show in Appendix A.4 the latency experienced by users increases in line with the number of simultaneous clients. Fortunately, with both BdbServer and BdbServer++ the requests, and the associated processing, are dealt with using batch queues on machines other than the gatekeeper. A significant number of users can simultaneously submit jobs without the performance of the system degrading significantly. We believe that there are unlikely to be hundreds of simultaneous users submitting jobs given the size of the collaboration and the number of sites available.

4.7 Conclusions

The BdbServer++ project taught several important lessons. There were delays in the project because we were attempting to develop a Grid application before the necessary Grid services had been developed and deployed. In retrospect, while the BdbServer++ project succeeded in its aim of investigating how the BaBar experiment could move towards a Grid architecture, the project itself could have been completed in far less time, and with less man power, if we had waited a year for the appropriate services to be developed.

The point at which we started development of the BdbServer++ software was, in retrospect, incorrect. During the development of BdbServer++ the experiment was moving from the CM1 to CM2 computing model. While the front-end of the system could be re-used, the packages which were used for data location and extraction would require modification to take account of the new data format.

Security awareness is an area of major weakness with the current implementations of the Grid. The move from computers being a single system, to networks of machines, and the issues this raises, have not been sufficiently addressed and are poorly understood by the general community. Anecdotal evidence from various discussions suggests that security, when it is considered, is not given sufficient priority, and is often added as an afterthought rather than as a core issue.

Because of this, we recommend that all software projects discuss their intentions with security professionals as part of their design phase. From our experience the criticism and feedback provided is useful. For core components, such as authentication and authorisation, which can and should be reused between projects, their advice can reduce development time, while improving the security and reliability of the final product.

BdbServer++ listened to this advice and made use of the work done by the Public Key Infrastructure projects which were running in parallel in the US and Europe for authentication and authorisation. We attempted to provide accounting through the use of a database system for user's requests, as this functionality was unavailable in the generation of Grid technologies being used.

Many of the issues which we had problems with, such as web authentication and core Grid services, have now been addressed. In the next Chapter we take the knowledge gained from this project, and the BaBar experiment, and apply it to the ScotGrid Tier 2 centre for LHC computing.

Chapter 5

ScotGrid - a prototype Tier 2 centre

In this chapter we present an overview of the ScotGrid prototype Tier 2 centre, its role within the current LHC computing infrastructure, and the work we have done on the development of the Edinburgh component of ScotGrid over the past three years. The evolution of ScotGrid's resources is discussed as an example of the changing, and increasingly sophisticated needs of the user community. We present our work on benchmarking the Edinburgh resources, and how this has fed into our deployment of resources. Further details of our work in developing knowledge about the capabilities of the existing equipment is presented in Appendices A and B.

5.1 ScotGrid - A Context and Motivation

ScotGrid is the original prototype Tier 2 centre for LHC computing in the UK. Our experiences with ScotGrid have fed into the development of the Northern, Southern and London Tier 2 centres through the GridPP collaboration. The ScotGrid project has been at the forefront of Grid development since its start, and maintains a novel infrastructure which is described in more detail in the following sections.

In this section we present the significant mile stones ScotGrid has reached, based on our definition of four phases. These cover the initial funding proposal presented by the Universities of Edinburgh and Glasgow, the equipment upgrade necessitated as our understanding of the system increased, the addition of the University of Durham, and finally, the newly proposed Scottish Grid Service which will include

the University of Dundee.

5.1.1 ScotGrid within the LHC Computing Model

As we discussed in Sections 2.6 and 3.1.1 the LHC computing model is based on a multi-tiered, hierarchical system which uses Grid computing as the underlying paradigm. The tiers in this model represent the detectors at Tier 0, national computing centres at Tier 1, regional computing centres - which support several local institutes - at Tier 2, and individual institutions at Tier 3.

Figure 5.1 demonstrates this. CERN is at the centre with the experiments, surrounded by the national centres which provide the core computing functionality. Around this are the Tier 2 sites, including individual laboratories and regional consortia such as ScotGrid, acting as a buffer between the national centres and the university based researchers.

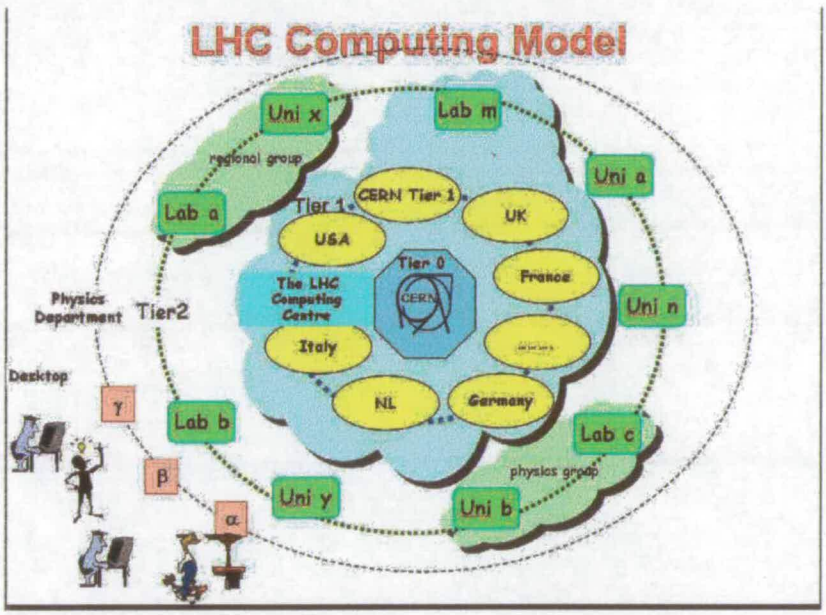

Figure 5.1: The LHC Computing Model

Within this model ScotGrid and the other Tier 2 centres have three computing roles. These are data storage, simulation production, and analysis. The sites can specialise in each of these areas to a greater or lesser degree. Later in this section we discuss how ScotGrid has balanced its resources as we have gained experience, and present our views on future developments in the conclusions.

5.1.2 The Initial Proposal

The ScotGrid prototype Tier 2 centre was proposed in early 2000 as part of the experimental computing infrastructure for the LHC. The secondary purpose of ScotGrid was to act as a testbed for Grid software within the UK. ScotGrid was intended to develop Scottish Grid experience for future projects both in particle physics and other disciplines, through the use of a novel computing infrastructure which made use of commercially available equipment and network bandwidth.

During the specification phase of the project the Universities of Edinburgh and Glasgow collaborated with SGI. A baseline specification was proposed which would support simulation production at Glasgow, and data storage facilities at Edinburgh. This was intended to provide a distributed system which could be regarded as a miniature Grid with each site specialising in a different area.

This was a divergence from the traditional development of such centres, where both computing and data storage resources had been located physically close to each other to reduce latency and simplify management. In ScotGrid's case it was decided to distribute the resources to discover how such a centre would function using commercially available network bandwidth rather than dedicated links. We discuss how successful this has been in later sections.

The specification assumed that the role of Tier 2 centres would be to provide processing for simulation production or specific analysis, and specialised services such as high performance graphics and storage. This was intended to be in the region of 10-20% of the storage and computing facilities of a Tier 1 site. The specification and road map assume that when the LHC starts actively taking data in 2007, ScotGrid will have the equivalent of 2500 PC99 processors (750 GFlops peak performance) and 100TB of storage capacity.

To start addressing this, SGI proposed that Edinburgh would host one of the first SGI Scalable Node Intel Architecture (SNIA) machines with 16 processors which would manage a 5TB RAID array. This would give Edinburgh 5% of the total storage required and a system which would allow development and training without an excessive initial outlay. This capacity could be increased as necessary.

The equipment envisaged for Glasgow included a 128 processor Beowulf cluster, equivalent to 250 PC99 systems or 10% of the requirement. As both physics analysis and simulation production are implicitly parallel the Beowulf architecture was an obvious choice as local network performance is not a limiting factor. It was assumed that data would be transfered between Glasgow and Edinburgh during quiet periods, so a facility was required to cache data at Glasgow during production. A 1.2TB RAID array was seen as being sufficient. Several graphic workstations were also proposed for testing advanced visualisation techniques.

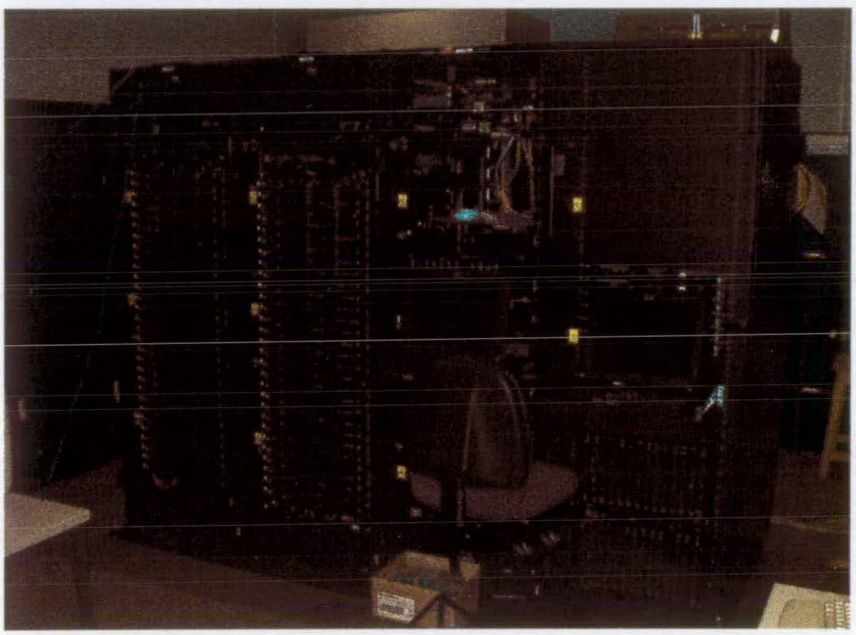

Figure 5.2: The Glasgow ScotGrid resources

The funding proposal was submitted in May 2000 to be considered as part of the Joint Research Equipment Initiative (JREI). The request was successful but, after competitive tendering, IBM was awarded the contract.

The system from IBM comprised a computing cluster with management nodes for Glasgow, shown in Figure 5.2, and a quad processor machine with a 5TB RAID array for Edinburgh. For historical reasons, and as a comparison to the current configuration which is presented in the next section, we present a summary of the major components of the original system. Details of cabling, racking and other sundry items can be found in [64].

Initial Edinburgh Resources

- IBM eServer xSeries 370

  4 × Intel PIII Xeon 700MHz processors

  16GB SDRAM memory

- 5TB RAID array

  FAStT500 RAID Controller

  70 × 73.4GB IBM HDDs (5TB)

Initial Glasgow Resources

- Storage Management Nodes

  3 × IBM eServer x340 dual Intel PIII 1GHz processors

- Compute/Head Nodes

  2 × IBM eServer x340 dual Intel PIII 1GHz processors

- Compute Nodes

  59 × IBM eServer x330 dual Intel PIII 1GHz processors

  59 × 2GB SDRAM

  59 × 20.4GB HDD (1.2TB)

Using these systems, both Edinburgh and Glasgow were able to acquire host certificates from the UKHEP Certificate Authority. After installing version 2 of the Globus Toolkit both sites were able to join the initial GridPP UK Grid monitoring map. The need for several new services was discovered at this point and projects were started to address them at a national level including virtual organisation membership lists and file management.

5.1.3 Phase 2 - Solving Initial Limitations

Once the IBM system had been installed, and was actively producing simulation data, it became obvious that there were two limitations. Firstly, the data storage capabilities at Glasgow were adequate for simulation production, but were insufficient for the intended data analysis. If physicists used Terabyte data sets, as the

LHC computing model intended, then the system could only support a single user's analysis no matter how significant their processing requirements.

The second limitation with the system was the lack of processing capability at Edinburgh. The original specification called for Edinburgh to provide only the data storage facilities and processing limited to data analysis. As experience with the system grew, it became clear that it was more efficient to process data near its storage location than move or copy data over the network to be processed. This is due to the different growth rates of processing, storage and networking. Since the growth rate of processing capability is significantly higher than for the others it is easier to install additional processing capacity where needed. The lack of processing capability in Edinburgh is clearly shown in Section 5.3.1 where we discuss the issues arising from installing the LCG2 software.

This situation was resolved by the purchase of a new storage system and several limited capability computing nodes which are described below. The storage system, which had been at Edinburgh, was moved to Glasgow giving them 5TB of RAID storage. One of the machines from Glasgow, a dual processor x340 system (Glenmorangie), was moved to Edinburgh in early 2003. The new system gave us three nodes which could be used as a buffer between the RAID storage system and the outside world.

Current Edinburgh Resources

- 2 × IBM eServer xSeries x330 (Glenlivet and Glenellen)

  1 × Intel PIV 1.8GHz processor

  256MB memory

  35GB HDD

- 1 × IBM eServer xSeries x440 (Glenkinchie)

  8 × Intel Xeon MP 1.9GHz processors

  32GB memory

  2 × FAStT900 RAID Controllers

  340 × 70.5GB HDDs (24TB)

The new Edinburgh system is shown in Figure 5.3. Of the three units, the near two house the disk drives, with the farthest one housing the compute nodes. This system is currently installed at the University's Advanced Computing Facility.

Figure 5.3: The Edinburgh ScotGrid resources

5.1.4 Phase 3 - Adding the University of Durham

The University of Durham officially joined the ScotGrid collaboration in September 2004. Researchers and staff at the university had been active for some months prior to this in co-ordinating activities with the two initial sites, through the Technical and Project Management Boards, to gain experience and decide on future activities. Figure 5.4 presents a logical view of the ScotGrid system.

As can be seen in Figure 5.4, each of the sites now has Grid front-end machines positioned between the site's resources and the Internet. These front-end machines provide specialist services and allow experimentation with the software used for the Grid interfaces without risking the resources they represent. Using this configuration each of the sites has a "well known" system that requests can be submitted to, and which can manage local resources. This allows administrators to remove machines from the Grid for maintenance without the users being aware of the change in resources.

ScotGrid Durham

The resources Durham brings to the ScotGrid collaboration includes a 40 node cluster. Each of these nodes have dual-2.2GHz Pentium 4 processors, 2GB of memory

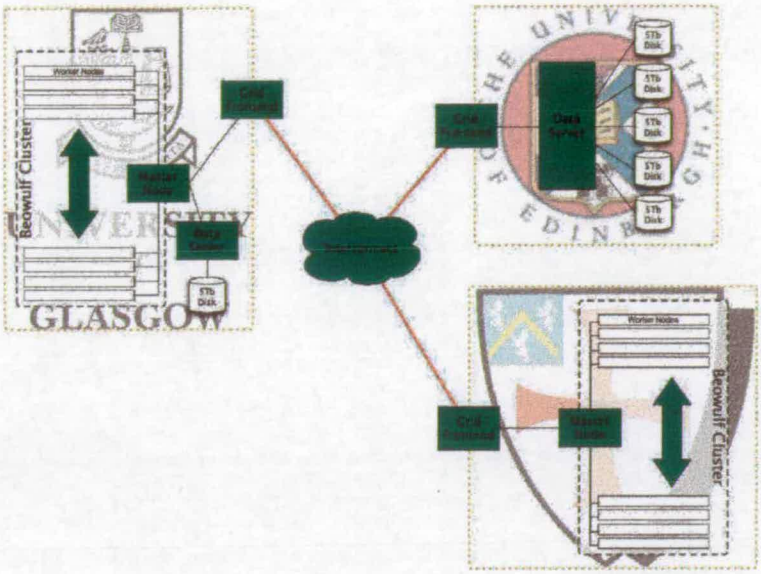

Figure 5.4: Abstract model of the ScotGrid Tier 2 centre

and 30GB of local disk. Durham also have a 5TB RAID array which, like Glasgow's is used to buffer data produced by the cluster before transfer to Edinburgh or to cache data for analysis.

5.1.5 Phase 4 - A Scottish Grid Service

The Universities of Edinburgh, Glasgow and Dundee are currently in the process of bidding for new funds to create a Scottish Grid Service as part of the UK National Grid Service [57]. This will be complimentary to the work of ScotGrid as the National Grid Service is intended to provide generic computing and data storage capabilities which can be used by a range of disciplines.

It is hoped that this Scottish Grid Service could be created and operational in time for the start of the LHC in 2007. This would provide us with access to more resources, and more significantly, allow us to test the portability of the analysis and simulation software for the LHC experiments. ScotGrid would also benefit from the ability to test the capabilities of resource allocation managers over multiple domains and varying allocation schemes.

5.2 Benchmarking ScotGrid Edinburgh

To make full use of the resources available at Edinburgh it is necessary to know what they are capable of in an existing framework. To this end we have conducted several experiments, the results of which are summarised in the following sections and more fully explored in the Appendices. The tests we conducted are not exhaustive but are intended to provide a guide for future work on the system, and to aid decision making as we expand the capabilities.

5.2.1 Globus Grid Interface

In Appendix A we present the results of the tests we conducted on one of the Edinburgh ScotGrid machines to discover the performance of two versions of the Globus 2.x Toolkits. These tests were conducted to allow us to gauge the effect of increasing the computational complexity of Grid proxy certificates, and the effects of increasing numbers of users submitting requests to a single resource. This was done as part of our work on BdbServer++ (Section 4.5) which was discussed in the previous chapter.

Figures 5.5 and 5.6 show the increase in User and Real time for the two toolkits tested as the proxy key length is increased. From these, we can see immediately that the real, or total, time taken to complete a request has decreased dramatically between the two releases from nearly 7 seconds using 2.1.0 to just over 1 second with 2.4.3.

Looking at the results of the average user time taken for each request however, clearly shows that this performance improvement is due to improvements in the Globus server and gatekeeper software rather than in the client utilities. Here the aggregate time of the user and system has remained constant at 0.3 seconds per request when using a 512 bit proxy.

In Figures 5.7 and 5.8 we present the results of our tests on the effect of increasing the number of clients submitting requests to a single machine. As we can see, there is an almost linear relationship between the number of clients simultaneous submitting requests and the completion time.

Our conclusions from these tests are that the Globus software has improved dramatically over the lifetime of the 2.x series of releases, and that we will require

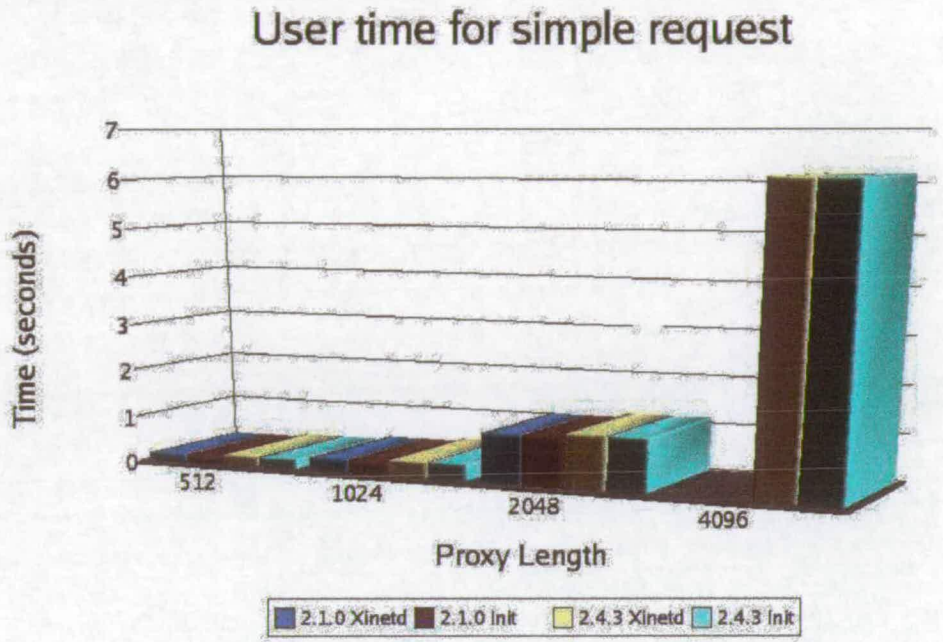

Figure 5.5: Comparison of User Time between Globus Versions in seconds

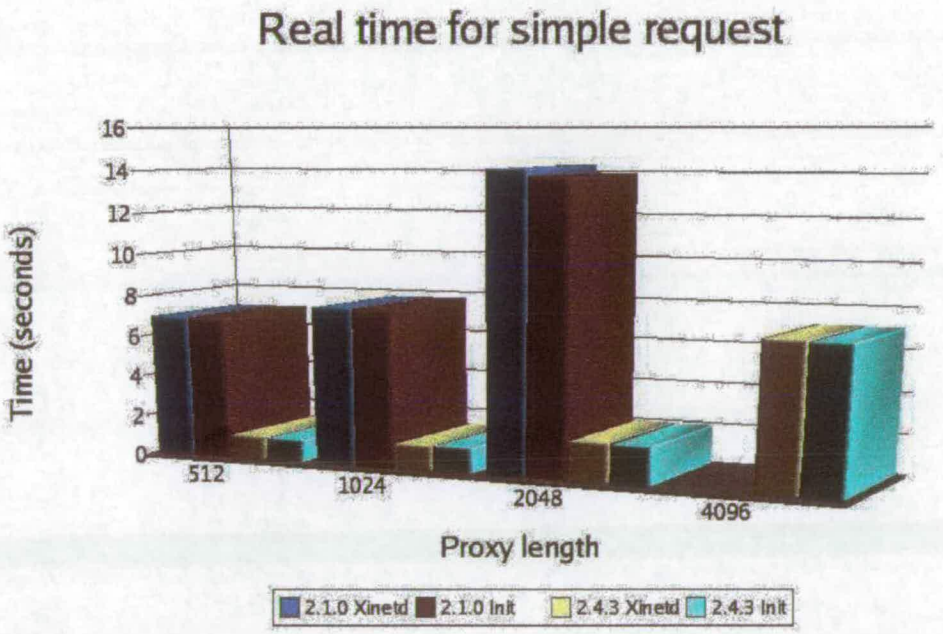

Figure 5.6: Comparison of Real Time between Globus Versions in seconds

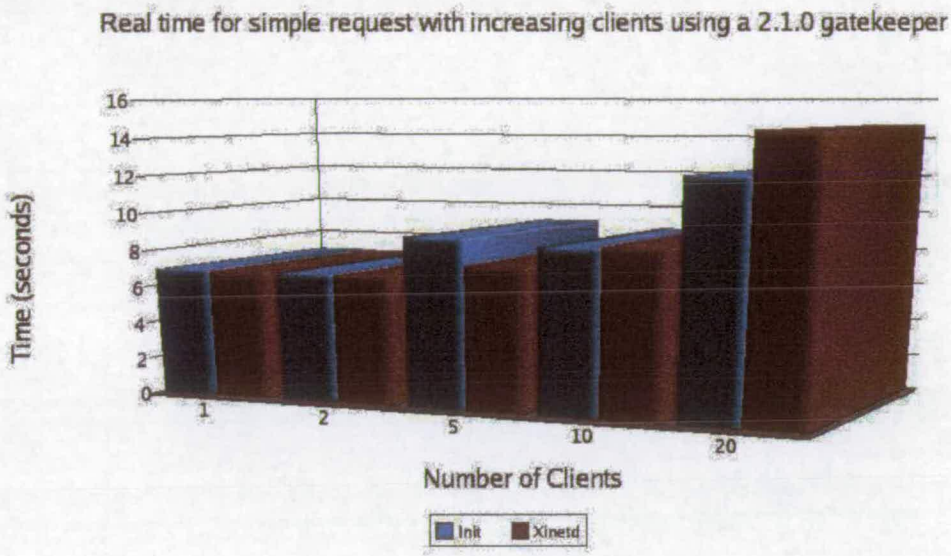

Figure 5.7: Effect on real time when increasing the number of concurrent users of a 2.1.0 gatekeeper in seconds

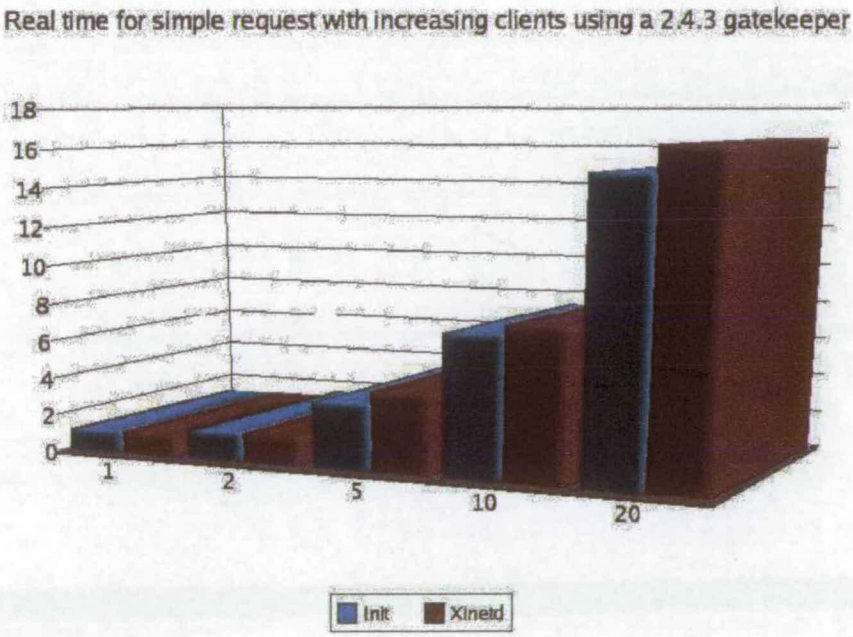

Figure 5.8: Effect on real time when increasing the number of concurrent users of a 2.4.3 gatekeeper in seconds

one gatekeeper machine for every 50 simultaneous requests, irrespective of computational resources. This is currently useful as the LCG software is based on the Globus Toolkit 2.x releases, but will require revisiting when particle physics adopts web services and either the Globus 3.x or 4.x code base.

5.2.2 Computing Capabilities

Appendix B presents an extended version of the results obtained from running the High Performance Linpack benchmarking software on Glenkinchie, our eight processor machine. Linpack is a widely used benchmarking suite which performs a variety of linear algebra and matrix calculations to discover the number of floating point calculations per second the system can perform. Linpack is used to create the Top500 list, which catalogues the worlds most powerful computers. Our tests concluded that the system currently ranked 500th has only 80 times[1] the processing capability of our system!

We conducted tests on Glenkinchie using a variety of libraries including the Automatically Tuned Linear Algebra Software (ATLAS) and Goto libraries. These tests looked at system performance using 2, 4 and 8 processors and problems of various sizes from the trivially small to ones which taxed the full scope of the system's resources. From these tests we reach the general conclusion that system performance improves as the size of the problem set is increased but that the performance per processor decreases as more are added. This is shown in Figure 5.9.

Although Glenkinchie has 32GB of memory, none of the tests were able to address more than 2GB per processor. With the eight processor system we were unable to achieve more than 16GB in total. The maximum matrix size achieved of 45,000 using an 8 byte value, results in 45,000 $\times$ 45,000 $\times$ 8 bytes = 16.2GB. This is, however, greater than the 2.15GB possible with 32 bit addressing with parity (i.e. 31 bit addressing). In addition to the Linpack software, other processes were in operation with their own memory requirements.

The Intel Xeon MP processors used in the system support 36 bit addressing (64GB per processor) but we are unable, based on our current knowledge, to conclude whether our inability to address the full scope of the system memory was a Linpack

[1] November 2004 list

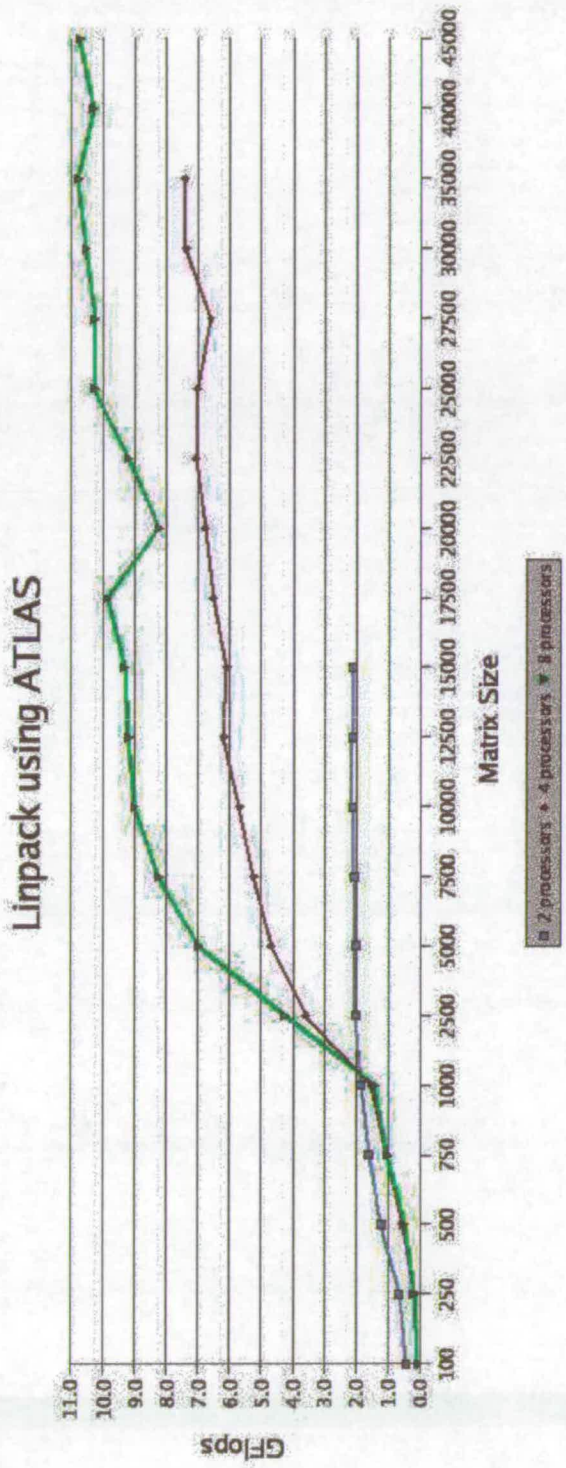

Figure 5.9: Results of varying processor numbers using ATLAS

or operating system issue. This question may be answered with the scheduled upgrade to the operating system.

Monitoring of the processes running on the system using the *top* utility show that the Linpack software was only able to use seven of the processors fully, with the eighth only running at around 50% utilisation. The remaining capacity went primarily to memory management software and operating system functions. The effect of this is discussed in Section B.4.3.

In none of the results we have presented have we included error bars. If this were an exhaustive test, which was intended to prove or disprove theories, then this would be an issue. However, with computer systems we accept that there are a large variety of factors affecting performance, including the hardware configuration, operating system version, load, and software versions used. We can therefore say that these tests have shown trends in the system performance of Glenkinchie, but should be regarded with caution as absolute statements.

5.2.3 Storage Performance

We conducted tests to discover the performance of the current RAID system using the Threaded IO [77] benchmark. These tests were initially conducted on one of the RAID arrays mounted as a local file system on Glenkinchie and then as an NFS mounted partition on Glenmorangie. The results provide us with basic information about how the system copes with varying file sizes and numbers of threads, or clients, in terms of its read, write, random read and random write performance.

Threaded IO was used to conduct tests using small file sizes ($\leq$100MB) and between 2 and 20 clients. Figure 5.10 shows the performance per thread for the array when mounted locally and remotely. For small file sizes and a small number of simultaneous clients, the system performs better over NFS than it does as a local file system. This is obviously due to file caching by NFS as it is clearly impossible to achieve this level of performance given the existing network hardware which is discussed in the next section.

As the number of clients increases the performance of the array mounted using NFS declines, relative to the results when locally mounted. When using 20 clients, the

NFS performance is roughly half that of the locally mounted system. This is logical, as the physical performance of the disk array and network bandwidth, shown in the next section, are fixed.

Figure 5.11 shows the file read performance of the partition when mounted over NFS using a 100MB file for each client. As we can clearly see, between 4 and 16 clients the variation in the performance, as shown by the error bars, is significant. This is due to caching by NFS where recently accessed files can be retrieved at a far higher speed than uncached ones which must be copied over the network. With 16 or more clients there is a distinct drop in aggregate performance from approx. 400MB/sec to under 100MB/sec. We attribute this to the configuration of the maximum number of simultaneous NFS connections on Glenmorangie.

Figure 5.12 shows the file read performance of the partition when it is locally mounted. It clearly shows a far lower deviation in the results and a more consistent data rate even with a larger number of clients. We discuss the effect of this in terms of system management in the Conclusions to this chapter.

Having conducted these initial tests we moved to using the IOZone benchmark. IOZone provides a wider range of tests than Threaded IO and can be used to discover both the performance of NFS file systems and simulated database systems. This is of interest to ScotGrid, as we have a researcher at Edinburgh working on the LHCb Conditions Database, and knowledge of how our existing hardware would perform as a database server would be useful.

These results were consistent with those of the Threaded IO tests. In Figure 5.13 we show the read performance of a partition mounted over NFS on Glenmorangie using a range of file sizes between 1KB and 64MB. We can see the same decline in performance experienced with Threaded IO. In Figure 5.14 we show the write performance which exhibits the same rapid decline in performance as file size increases and network bandwidth becomes the bottleneck.

Figure 5.15 shows the re-read performance achieved with IOZone. The performance declines as the file size increases and its likelihood of being retained in the cache decreases. Figure 5.16 shows the large variation in performance for re-writing to small files. Again, the network performance appears to be the limiting factor with

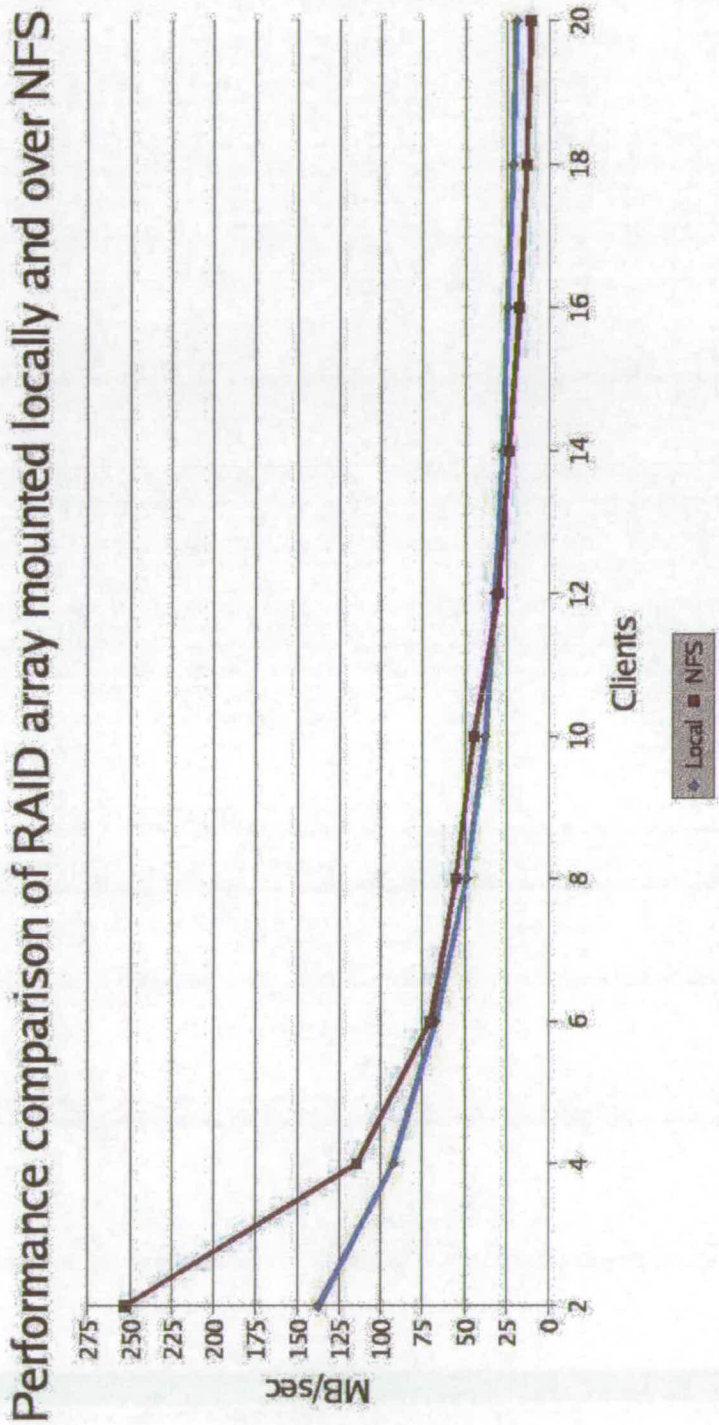

Figure 5.10: File read performance (per client) of RAID partition as local file system and NFS file system in MB/sec

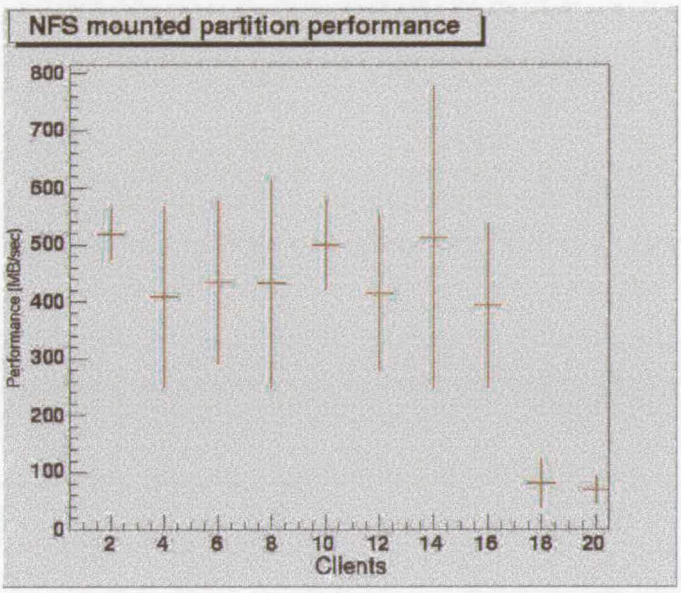

Figure 5.11: Read performance for NFS mounted RAID partition in MB/sec using 100MB file

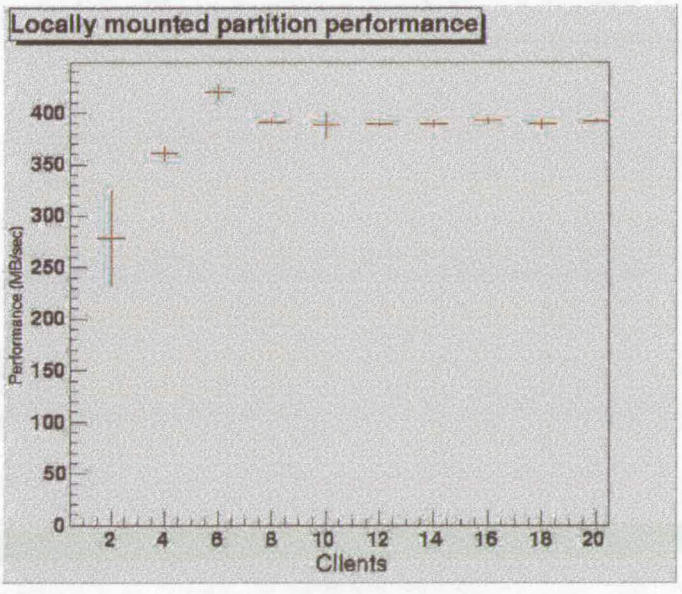

Figure 5.12: Read performance for locally mounted RAID partition in MB/sec using 100MB file

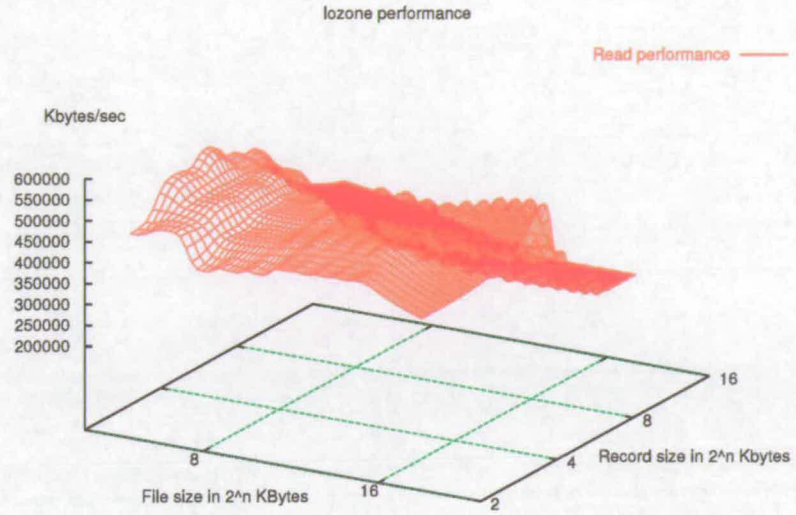

Figure 5.13: Read performance for locally mounted RAID partition using the IOZone benchmark

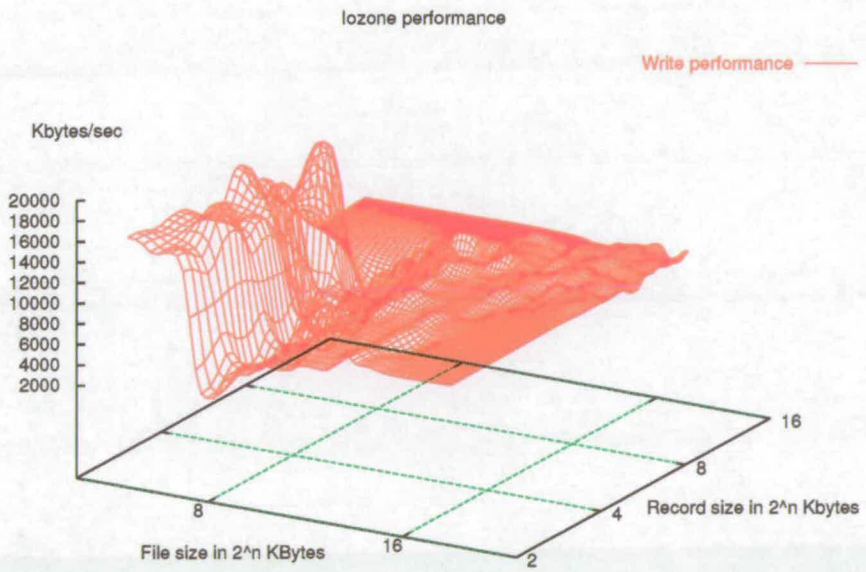

Figure 5.14: Write performance for locally mounted RAID partition using the IO-Zone benchmark

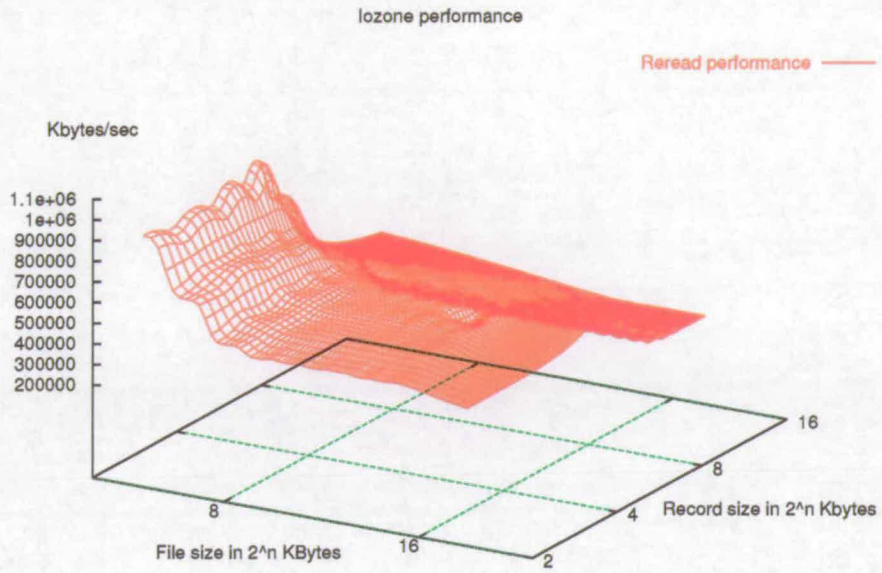

Figure 5.15: Re-Read performance for locally mounted RAID partition using the IOZone benchmark

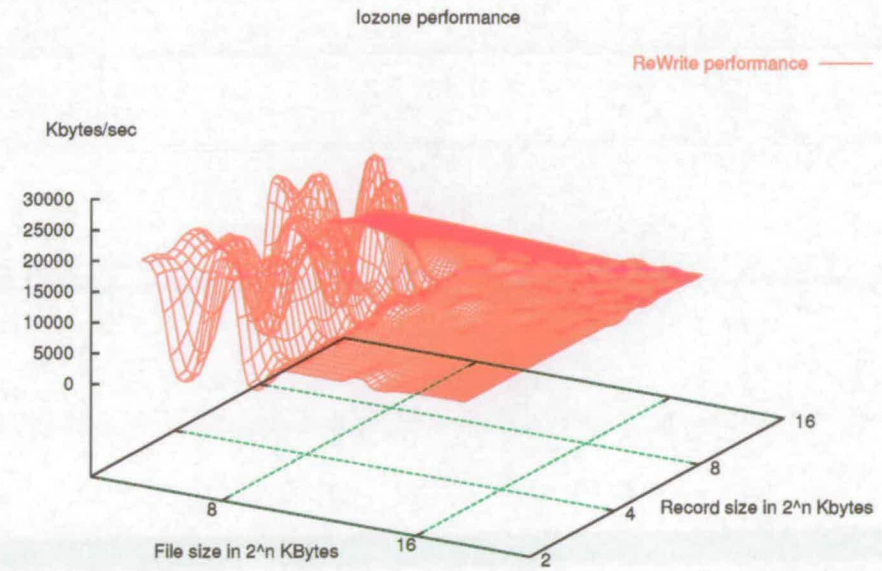

Figure 5.16: Re-Write performance for locally mounted RAID partition using the IOZone benchmark

large file sizes.

5.2.4 Network Performance

Each of the three sites in ScotGrid are connected over the SuperJanet network, and are therefore at the mercy of other system traffic reducing our available bandwidth. However, it is possible to test the performance of these links to gain an understanding of the available bandwidth, and use this to plan for future development.

We tested the various connections with a simple network traffic generator (stackthru [76]). This package comprises a listener program which is run on the server and a client. This generates a specified quantity of random data in a proscribed block size, which is then transfered to the server using either the TCPv4 or UDPv4 protocol.

Using this package with TCP we can discover the actual network bandwidth between the two machines, as it is a reliable transfer protocol, i.e. it ensures that data is correctly received. With UDP we cannot guarantee the data is received by the server, but can see the maximum performance we can expect from the processor and network card on the client machine. This is important as we move towards Gigabit networks, where there is significant processor overhead required to achieve this level of throughput.

Each of the tests in this section were conducted 20 times to achieve a statistically meaningful sample without overloading the network.

Using stackthru we first tested the connection between Glenlivet and Glenkinchie on the SRIF network, shown in Figure 5.17. These two machines are connected through a single switch using Gigabit fibre and copper connections. We would expect data transfer to achieve a theoretical maximum of 62.5MB/sec, or more realistically 56.25MB/sec assuming a 10% overhead in half-duplex mode and 125MB/sec in full duplex mode..

Table 5.1 shows we achieved a maximum data transfer rate of approximately 91%, using TCPv4 of the theoretical limit of half-duplex with our existing equipment. The results of the UDP tests are less than expected and will be discussed in relation to the results achieved at other sites later. The high variance in the results,

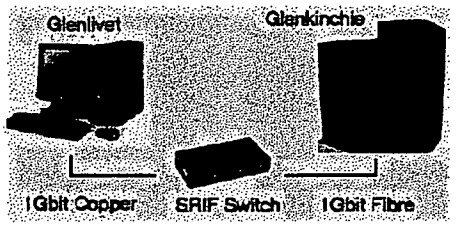

Figure 5.17: View of the Glenlivet - Glenkinchie connection

especially for the UDP tests, indicate that even in a controlled environment there are external factors such as CPU load, which can significantly effect performance.

These results provide reassurance that outside a controlled and known environment the figures produced will remain useful. For the next stage we conducted the same test between a machine (Blacksheep) on the Physics network and Glenlivet. Based on the results of the *traceroute* program this connection involves three hops with departmental, university and ScotGrid routers between the two machines as shown in Figure 5.18.

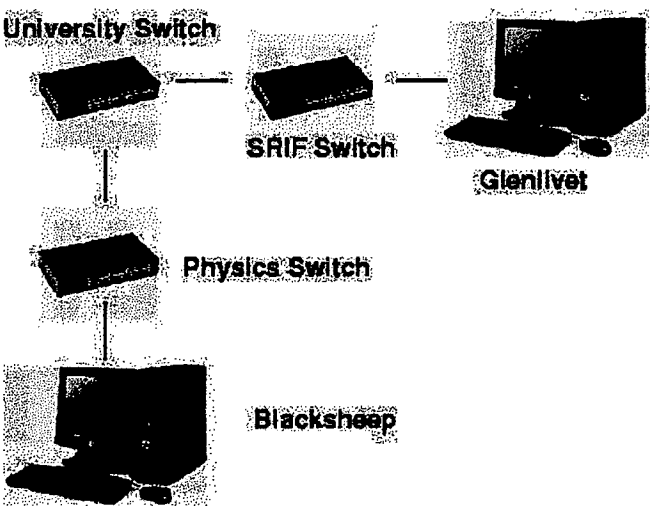

Figure 5.18: View of the Blacksheep - Glenlivet connection

We know that the Physics department machines are currently using 100Mbit copper connections. Assuming that this is the slowest part of the network, we would expect to achieve a theoretical maximum of 6.25MB/sec or with a 10% overhead 5.625MB/sec using half-duplex. Table 5.2 shows the results we achieved. The results

Transfer Size	1MB	5MB	10MB	15MB	16MB	20MB	25MB
TCP Ave. Throughput	56.59	55.72	55.83	54.88	55.51	52.17	53.73
Standard Deviation	2.38	0.85	4.5	6.61	3.28	9.87	6.19
UDP Ave. Throughput	93.18	91.92	80.18	90.13	89.92	89.89	88.81
Standard Deviation	4.04	0.85	21.92	5.35	6.63	5.76	7.43

Table 5.1: Data throughput between Glenlivet and Glenkinchie in MB/sec for varying transfer size

Transfer Size	1MB	5MB	10MB	15MB	16MB	20MB	25MB
TCP Ave. Throughput	10.34	8.55	9.23	9.84	10.19	10.29	10.32
Standard Deviation	2.31	1.76	1.35	1.62	0.63	0.85	0.88
UDP Ave. Throughput	11.34	11.39	10.33	10.3	9.71	10.35	10.15
Standard Deviation	1.52	0.69	2.07	2.43	2.5	1.98	2.01

Table 5.2: Data throughput between Physics and SRIF network in MB/sec for varying transfer size

Transfer Size	1MB	5MB	10MB	15MB	16MB	20MB	25MB
TCP Ave. Throughput	1.85	1.41	1.39	1.27	1.15	1.17	1.14
Standard Deviation	0.97	0.14	0.06	0.07	0.09	0.08	0.12
UDP Ave. Throughput	100	98.33	98.57	98.46	98.49	98.4	98.22
Standard Deviation	0	0.72	0.5	0.32	0.37	0.21	0.84

Table 5.3: Data throughput between Durham and SRIF network in MB/sec for varying transfer size

Transfer Size	1MB	5MB	10MB	15MB	16MB	20MB	25MB
TCP Ave. Throughput	16.69	21.6	22.48	21.19	21.58	21.02	21.58
Standard Deviation	3.45	3.66	0.57	3.26	2.14	2.98	2.4
UDP Ave. Throughput	101.11	98.92	98.72	98.43	98.46	98.22	98.46
Standard Deviation	3.42	1	0.46	0.32	0.31	1.13	0.12

Table 5.4: Data throughput between Glasgow and SRIF network in MB/sec for varying transfer size

Transfer Size	1MB	5MB	10MB	15MB	16MB	20MB	25MB
TCP Ave. Throughput	0.79	0.82	0.83	0.83	0.82	0.81	0.81
Standard Deviation	0.07	0.02	0.01	0.01	0.01	0.02	0.02
UDP Ave. Throughput	1.19	1.14	1.14	1.14	1.14	1.14	1.14
Standard Deviation	0.01	0	0	0	0	0	0

Table 5.5: Data throughput between CERN and SRIF network in MB/sec for varying transfer size

for the TCP data transfers are twice our expectations. This confirms that Physics machines are configured to use duplex by default. The reason for this difference in configurations is due to the different intended uses for the machines. While the ScotGrid machines are expected to have multiple connections transferring data to and from them, the Physics machines are intended for single user usage. It therefore makes sense from an administrative perspective to configure the network connection to achieve either maximum data transfer, in the case of Physics, or maximum connections, in the case of ScotGrid. One other result of this test was a swift response from the Physics department system administrators, whose software had noticed the sudden increase in network traffic and started enquiries into whether an intruder had gained access.

With the experience gained from running this test between the Physics and SRIF networks, we spoke in advance to the system administrators at Durham, Glasgow, and CERN before conducting these tests with machines on their networks. In all of these cases there is no direct connection between the sites and the SuperJanet network must be used.

We also attempted to perform these tests using a development machine at RAL. After making a request, and following this up, after six weeks we were still no closer to being able to gain access to a machine.

The expectation is that both Glasgow and Durham, which have over 5TB of available disk, will cache data produced by their clusters during the day, and transfer data to the Edinburgh storage node during off peak hours, so that normal network traffic will not be disrupted by the large data transfers. This is also likely to be the method used by CERN. We therefore conducted these tests outside normal working hours, partly to achieve a higher data transfer rate, but mainly to ensure we did not swamp the network and disrupt other users work.

Using *traceroute* again, we identified the typical number of hops between sites as being: from Glasgow, 9; from Durham, 12; and from CERN, 15. This can change given network conditions, for example if there is a failed node and communications are rerouted, but can typically be estimated given a rudimentary knowledge of the SuperJanet network and University network structures.

While we managed to achieve a respectable 20MB/sec from Glasgow to Edinburgh (Table 5.4) the connection between Durham and Edinburgh (Table 5.3) shows obvious problems. One of the early suggestions for ScotGrid was to have all the data storage facilities at Edinburgh and only processing at the other sites. While our connection to Glasgow suggests we could sustain data transfer comparable to a single disk drive, the connection to Durham is more akin to the performance of a floppy disk drive.

The throughput using the connection from CERN (Table 5.5) is significantly less than from Glasgow. This may be partially, but not totally, due to the low specification of the CERN machine. The machine used is only equipped with a 10Mbit network card, as shown by the UDP results. However, even with this limitation the TCP results are significantly less than realistic expectations. The results we achieved were in the 800 KB/sec range with minimal variance (0.01-0.04) which is only 60% of peak performance.

The results of these tests identified a significant problem - the connection between Durham and Edinburgh was providing less than 1KB/sec of bandwidth using GridFTP. This compares to the 20MB/sec connection with Glasgow. The results from our TCP tests showed bandwidth in the 1MB/sec range. Further investigation showed that when transferring data using the HTTP protocol, we experience bandwidth of 1-3MB/sec and similar performance with the *scp* utility. We return to this in our conclusions.

5.3 LCG Experiences

In this section we present the components of the current LCG software, and the experience we have gained at Edinburgh from its installation and use. We discuss the experience our system administration staff gained from the installation of the software, and the experience we gained from walking a physics PhD student through performing analysis using it.

5.3.1 Systems Administration

The LCG software was installed at Edinburgh by Steve Thorn to replace the Globus and piecemeal approach we had installed and previously used. Installation was

conducted by configuring an LCFGng server on Glenellen which then installed the required operating system and software on the Compute Element (Glenlivet), token Worker Node (Ardbeg) and Storage Element (Glenmorangie). In this section we present the issues which were faced and our work to alleviate them.

The first issue faced during installation was the choice of the 7.3 version of Red Hat Linux by the LCG developers. This is a consumer grade version of Linux and does not support many of the features of our eServer x440 (Glenkinchie) and the RAID arrays. As the LCG software is platform dependent, we used Glenmorangie as the SE, and connected it to the RAID array over NFS. The performance achieved providing storage over NFS, rather than locally, was discussed in Section 5.2.3.

The biggest problem for Edinburgh was the configuration of the Storage Element. The current LCG software limits us to a single directory for data storage with each VO using a subdirectory within this. Our RAID array spans 24 partitions, each of approximately one Terabyte. This is due to the limitations of the IBM software and hardware. To solve this we investigated several options.

We initially created multiple mount points on Glenmorangie, and then used soft links from these to the */storage* directory which LCG uses. This allowed us to increase storage from one Terabyte for all VOs on the SE, to one Terabyte per VO. As we showed in Section 5.2.3 as the number of connections or clients increase, the file access performance over NFS degrades. We could avoid this by using separate storage elements for each VO. However, this would still limit each VO to one Terabyte and does not necessarily represent a sensible investment for performance enhancement but may for simplifying management.

We then investigated the potential of changing the file system from the current *ext3* format to another, which would allow multiple partitions to be presented as a single system. *ext3* had been chosen as it provides journaling, which reduces the time needed to restart in the event of a systems failure. We looked at several file systems which we discounted, because they were either inappropriate for our expected requirements, or would have resulted in the loss of data and downtime. We are currently investigating using dCache, an SRM compliant system, and POOL to provide access to the RAID.

Generally our administrative experiences with the LCG software have been acceptable. Some outside assistance from Tier 1 and other Tier 2 sites has been required, but to no greater extent than when we have installed other particle physics software. The experience with LCFGng at Glasgow and Durham, where there are significant numbers of identical machines, has been positive. We believe the availability of lightweight configuration tools which can rapidly reconfigure part or all of a cluster, will soon become important.

Edinburgh's situation is different due to the storage bias of the site. This has manifested itself in the higher configuration time per system, compared to other sites. This is something which has been fed back to the developers, as part of the call for a more selective and light weight installation system.

5.3.2 Physics Analysis

The purpose of the Grid is to conduct scientific research not just to test computer science theories. Moving to a new architecture does however, present new issues which must be addressed before wide deployment and use. To aid the migration of the physicists at Edinburgh we used one of the physics PhD students as a test subject to discover and document these problems. In this section we present the stages we went through, and the problems encountered, in getting to the point where he could perform analysis over the Grid.

We started by generating a grid certificate for the student. This raised several serious questions about the support available from the UK national certificate authority, their ability to deal with requests, and their internal communications. We accept that providing a national certification authority, which supports multiple disciplines and usage scenarios, is interesting from a practical perspective, and have fed our comments on the current situation back to the Grid Support Centre and their local representatives.

National certificate authorities are an issue worthy of further consideration and development. Unfortunately this is outside the scope of this thesis, but we believe there is much potential for improvement at both a practical level and from a research perspective.

Registration with the LHCb virtual organisation [47] was straight forward and accomplished within 24 hours, as collaborating sites access lists are automatically updated. As we discussed in previous sections, given the design of the ScotGrid system there are few machines which could be used as front-ends at Edinburgh. We therefore do not have a machine dedicated to being a User Interface (UI).

We investigated installing the UI on a desktop machine but several issues prevented this. These included its reliance on Red Hat 7.3 - which is no longer support by the School of Physics and is not supported by Red Hat. The stand alone UI installation comprises 138 packages including at least five programming languages and multiple job submission systems, many of which we either have installed already, or have no need of. We therefore used a UI at the University of Glasgow for job submission.

We started by submitting simple jobs to the Resource Broker to test that the local configuration and the user certificate were installed correctly and that our understanding of the Job Description Language (JDL) was correct. These tests were successful and expanded our understanding of the system which allowed us to progressed to more complex scenarios.

We investigated running analysis jobs using ROOT over LCG. At present there is no easy way to do this [123]. We are in the process of investigating methods for providing the software over the network e.g. GASS with Globus; dynamically installing the software on nodes either from source or pre-built executables; and using current information systems and their ability to discover the software installed on worker nodes.

Physicists need a compelling reason to migrate from their current analysis methods, to a new system with a steep learning curve. The growing scarcity of resources at the traditional analysis sites, coupled with the availability of resources at the Tier 2 sites, means that from the physics perspective, there is a desire to invest effort in solving this.

5.4 Conclusions

The experiences of the ScotGrid project show that there are advantages to collaboration between groups and disciplines in a local and regional context. A clear

indication of this are the resources of ScotGrid, which have expanded past those that particle physics alone could have purchased, with the addition of resources from the eDikt group and bio-informatics community. The inclusion of the University of Durham, and the planned inclusion of the University of Dundee, show that once projects such as this reach critical mass, they provide benefits for a widening community.

In exchange for the resources these groups bring to the collaboration, we have provided experience and support to those groups whose computing requirements are expanding past previous levels, and who have limited experience with the Grid so far. This is likely to continue if the Scottish National Grid Service, which we discussed in Section 5.1.5 is funded. We hope that this experience and the SNGS can be used to extend the Grid to research groups throughout Scotland.

SRIF initiatives, such as the University Storage Area Networks, which have been purchased and installed at Edinburgh and Glasgow, also contribute to this collaborative experience.

The results from our benchmarking of the ScotGrid hardware highlighted several interesting issues. Because of our work, the problems with the Durham internet link, specifically the default port throttling of their firewall, were discovered and investigated. Performance testing of the RAID array, both locally and remotely mounted, provided us with the information needed to make an informed choice about how to manage the Storage Element. Because of the more consistent performance of the RAID array mounted on Glenkinchie, it was decided to attempt to move the SE installation to Glenkinchie which will free Glenmorangie to act as an additional worker node or even user interface.

The benchmarking of Glenkinchie using Linpack was interesting because it showed the level of performance gain we can achieve by tuning our software to a greater degree. It also emphasised that each of the tests we conducted are relatively self contained, and can be extended to a more detailed level if necessary. These experiments would make interesting MSc or senior undergraduate projects, as they involve system software, communication networks and physical hardware.

In the future we expect the ScotGrid system to go through changes to its hardware,

software and configuration between now and 2007. The data storage capacity of the system is currently less than half what will be required for the start of LHC, and the processing capacity also needs increasing. The likely changes in this area, both at Edinburgh and the other sites, is a move towards 64 bit processor architectures, which will allow processors to directly access far more memory than the 32 bit systems currently in use.

The experiences we have had with the LCG software have led to several conclusions. Based on our experiences with the BBGUtils package in Section 4.5.1, a lightweight User Interface which can be installed by the user will be necessary for wide deployment and use. The present heavyweight installation is impractical for the majority of users, unless their institution has decided to actively support the Grid. The documentation for this also needs to be improved.

Within the LCG software there are still operating system dependencies which will need to be removed, as it is unlikely that even an international laboratory will be able to dictate institutional level computing policy. These dependencies manifest themselves as kernel version specific issues, as well as application and utility version specific requirements. The concept of a sand-box for processing has merits. It would also assist Tier 2 system administrators, who are becoming concerned at the perceived loss of control of their resources to the Tier 1 centres.

Many of the issues for both the server software and the UI could be addressed by the development of selective installations. These would allow the installer to decide which components are added. This, coupled with a move of functionality to the middleware layer, which is where we believe it belongs, would remove the need for the bulk of the software in the current releases. Software such as the platform specific job submission clients included in the UI (Section 5.3.2), should be at the middleware layer, not the client level.

Chapter 6

Grid Data Storage

In this chapter we present the SRM2SRB project. This has investigated, designed and started development of a novel interface to an existing distributed storage system, the SDSC Storage Resource Broker (SRB), using the Grid's storage interface standard, the Storage Resource Manager (SRM). We consider the problem of multiple information systems within a Grid, and present a method of ensuring that prior development work need not be lost when moving between solutions.

6.1 Motivation

Our motivation for this project is to gain experience with generic data management systems and to address two questions. How are users able to gain access to the data they require if it is stored at a remote location? How can the information structures required for data location, management, and retrieval be made to interoperate between different solutions?

This is of interest to us because of the storage role Edinburgh plays in ScotGrid, as discussed in the previous Chapter. The storage resources at Edinburgh are expected to support multiple disciplines. We assume that these groups will use different storage management solutions, and that we will need to make these solutions interoperable, or discover some other way of managing them.

We have taken two existing distributed data management solutions, the SDSC Storage Resource Broker, and an implementation of the Storage Resource Manager specification, and investigated how we can make them interoperable with each other.

These systems were chosen because SRM is emerging as the dominant generic interface to storage systems for the Grid. SRB in contrast was a component of the original software distributed as part of the UK e-Science programme in 2001. There are a number of projects in several disciplines which have used SRB for a proof of concept. SRB has attributes which make it an ideal example of a monolithic and proprietary system.

6.2 A Brief History of the SRM2SRB Project

The idea for the SRM2SRB project started in early 2004. It was becoming obvious that SRM was gaining popularity at the international and American research laboratories, with the development of interfaces to their HSM systems and the subsequent widespread attention at Grid related conferences. Having done some previous work on an early release of SRB [122] I had some contact with their development group, and felt that it was worth discovering if it was possible and interesting to create an interface between the two.

I arranged a meeting at GGF10 in Berlin (March 2004) which included Owen Maroney (Bristol), Morag Burgon-Lyon (Glasgow), David Berry (NeSC) and Reagan Moore (SDSC). The GridPP2 funding had been announced and Edinburgh, Glasgow and Bristol comprise the three groups which will be supporting data management and storage in the run up to the start of the LHC. Dave Berry is the NeSC Visitor Programme manager and Reagan Moore is in charge of the SRB project.

As a result of this meeting we arranged for Wayne Schroeder, a senior developer with the SRB project, to visit the eScience Institute (eSI) for two weeks in May 2004. This was intended to give us an overview of the current state and plans for the project. An added advantage of the visit would be the provision of training through a seminar series for UK researchers.

Shortly after the SRB meeting I became aware of the work of Simon Metson at Bristol with the G-MCAT project (described in Section 6.3.3). This obviously has an impact on the project, and I discuss how it affects the design of the interface in Section 6.4.3.

To gain further knowledge about SRM I visited the development group at LBNL in early May 2004. This allowed me to discuss their implementation and ask for their suggestions about the concept, scope and usefulness of the project. Although we decided not to use the Berkeley implementation, its design is discussed in Section 6.4.3 as a comparison to the version we did use.

After the success of the SRB visit we were able to convince Don Petravick and Timur Perelmutov of FermiLab (FNAL) to take part in the visitors programme for a week in October 2004. We used the experience we had gained from Wayne Schroeder's visit to structure this week in a way which would allow the most number of people to benefit from it. We arranged a public talk, aimed at the general audience, for the beginning of the week. This provided an overview of SRM and how it fits into the FNAL infrastructure.

This led into a two day developer workshop where we installed and used the FNAL version of SRM and had a code walk through from Timur. This proved to be very useful for our development work as we show in Section 6.4.4. Development of the software was started in October and is currently ongoing.

6.3 Existing Software

In the following sections we present a very brief summary of the software used by the SRM2SRB project, and provide links to further information about them.

6.3.1 SDSC Storage Resource Broker

The SDSC Storage Resource Broker (SRB) [75] project was started in 1997 as part of the NPACI Data Intensive Computing Environments (DICE) initiative. SRB is a client-server system based on a three tier model which provides several different interfaces into distributed heterogeneous data resources, which may be spread across multiple administrative domains. SRB is a monolithic system which has several advantages over systems which attempt to build consensus between developers in different groups (see Section 6.3.2).

The SRB system provides an abstract or logical view of the data to which it has access. It provides high level containers, similar to directories in a file system, called

collections. Digital entities within a collection, such as files and database BLOBs, are not necessarily based on the same physical resource, but have been grouped together by either a user or some logical relationship. Each SRB system maintains a central metadata repository called the MCAT which stores information on resources, data and users in a relational database.

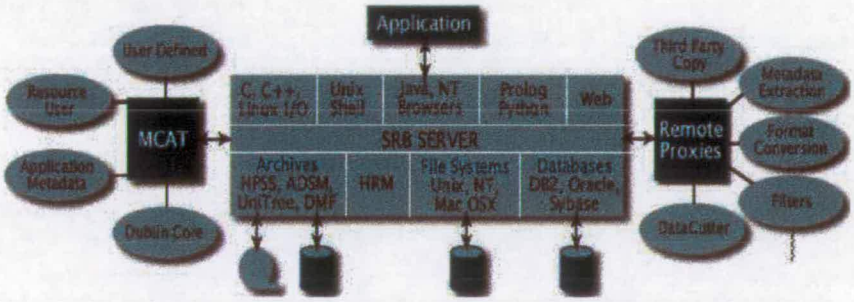

Figure 6.1: Overview of the SRB system

The Meta Data Catalogue - MCAT

The metadata catalogue, or MCAT [51, 52, 15] is the central data repository for each SRB system. The current release supports MCAT being installed on the IBM DB2, Oracle and PostGres relational database systems. In all cases it supports:

- The storage of metadata on data sets, users, resources, and proxy methods.
- Replica information for data and containers
- Global user name space and authentication
- Authorisation through ACL and tickets
- An audit trail on data and collections
- Resource transparency for data location and protocol negotiation

The SRB Vault

SRB replicates entities which have been flagged as archival data automatically. As it cannot store data of this type in individual user's account, as this could lead to data corruption, the system creates what is known as a Vault. The Vault is controlled by the SRB account on each SRB physical resource. Archival data can be written into the Vault by the system and extracted and copied from it to improve performance by the use of replication, redundancy and proximity to the requester's resources.

6.3.2 Storage Resource Manager

The Storage Resource Manager (SRM) specification [134, 135] has been under development for several years with active interest in both Europe (EDG and CERN) and the United States (LBNL and FNAL). It has been discussed at many sessions of the Global Grid Forum. Unlike SRB, SRM is not an application or software suite but a specification for how a data resource should present itself to the Grid, and the methods and responses its implementations should provide. It describes the abstract concepts behind the methods it recommends for implementing space management, data transfer, request status, directory and permission functions.

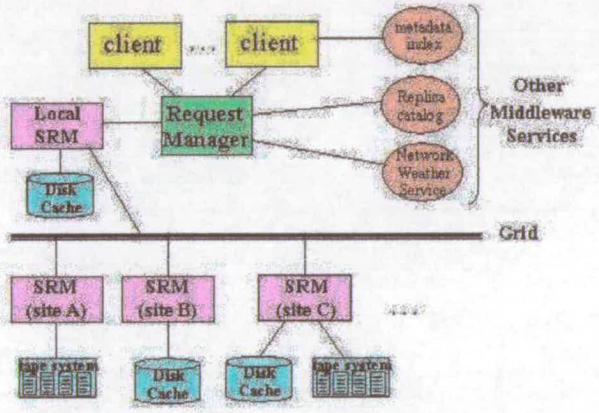

Figure 6.2: An SRM system

Implementations of SRM have been developed to provide interfaces to mass storage systems such as HPSS, Enstore, JasMINE, CASTOR, EDG SE and ATLAS, and for disk systems such as RAID arrays. Figure 6.2 demonstrates a conceptual SRM system, where the specification is used to provide access to disk and tape systems, while other services are used to provide location, routing, and request management.

Within SRM there are three types of storage: volatile, durable and permanent. Volatile storage is used for lifetime limited files, e.g. temporary files which are directly available online. Durable storage provides a middle ground between volatile and permanent files. These are not lifetime limited, but are expected to be either deleted or moved to permanent storage. Permanent storage is used for the long term and robust storage of files.

Data location

The SRM specification does not define how data is to be registered or located in the system, nor the security measures which should be used to provide access to the data. It does recommend protocol negotiation to allow clients and servers to agree on a data transfer protocol, such as GridFTP, which they both support. This solution allows projects to specify the attributes for their data rather than conforming to a single definition.

6.3.3 G-MCAT

The GMCat project [98] at the University of Bristol has developed a web application which registers a translated version of the metadata from the MCAT of an SRB system with the Replica Location Service (RLS) used by EDG and LCG. This is intended to provide information about the physical data locations of the SRB files but does not deal with the security and file access implications. These will be discussed in Sections 6.4.2 and 6.4.3.

6.4 The SRM2SRB Project

In this section we present information about the experience we gained from installing the software described in Section 6.3. We present our analysis of the problems this software could be used to address, and the scenario we regard as the most likely. Using this, we present the possible designs for the system based on the constraints of the existing systems, and comment on our progress so far in developing this software.

6.4.1 Installation

To gain a better understanding of the software, we installed file system SRM's from Berkeley and Fermi Laboratories and SRB version 3.2.1. We installed SRB on several machine connected to the Physics network using the PostGres 7.4.1 database for the MCAT. We documented these installations and the minor configuration issues which occurred. Using this we were able to successfully create multiple users, add files and replicate these to multiple remote machines.

As part of an MSc project Berkeley SRM was installed from source on a development machine connected to the Physics private network. Problems were found with

the installation in terms of the compiler version required, and the extra software packages needed. These were documented and passed on to the developers. An installation of the binary distribution using a RedHat 7.3 machine was successful, but as we show in the next section, the design and intentions of this software were incompatible with our needs.

As part of the visit by the FNAL SRM developers we created a local Certificate Authority for authenticating machines and users. Using these certificates we configured multiple machines with the Globus toolkit and installed SRM on top of this. We were able to successfully add, removed and manipulate files in this system.

6.4.2 Problem Analysis

From discussions with the SRB developers, and interested individuals including a member of the eDikt group, we defined three scenarios that the system could be used for. These are :

- predominately writes
- an equal mixture of read an write operations
- predominantly reads

With the first scenario SRB is used by the system for data storage and replication, and will continue to be used in the future. User applications can be developed based on either SRM or SRB interfaces making it more attractive for developers. CCLRC has done some development work in this area [119], intended to standardise their resources and present a common web interface for users. In terms of the metadata system, the primary issue will be to generate a unique file identifier and provide this to the user application.

The second scenario suggests that we have two equal, and fully interoperable, systems. These exchange data between each other freely, and have compatible metadata and information systems. While this would be an ideal long term situation, we do not believe that at the moment this is a realistic expectation. From the details we presented in Section 4.4 on data distribution for the BaBar experiment, there does not appear to be a particle physics data scenario which would fit this, although there

may be in other disciplines.

In the final scenario, which we regard as the most likely for particle physics, the SRB system is used to maintain the data of an existing project. Potentially this project may migrate to an SRM based system in the future. This option has been considered by the CMS experiment. To aid data location and retrieval the user must be able to provide the system with a data location (URI) which is comprehensible to both SRM and SRB. This is the problem the GMCat project is attempting to solve. We return to this issue in the next section.

Of these three scenarios we believe that the third is the most likely for the SRM2SRB project. As we have shown in previous chapters, particle physics experiments typically have one or a small number of data producers, and a much larger number of data consumers. In the next section we explain how we have developed a design for a system which takes this into account.

6.4.3 Design

Direct Access Design

We initially look at the design of an interface between an SRB data store and any Grid data retrieval system which uses the RLS for data location discovery. This is presented in Figure 6.3. In this figure we have GMCat installed at the same location as the MCAT and publishing information about the physical location of data, including the hostname and full path name, to the RLS.

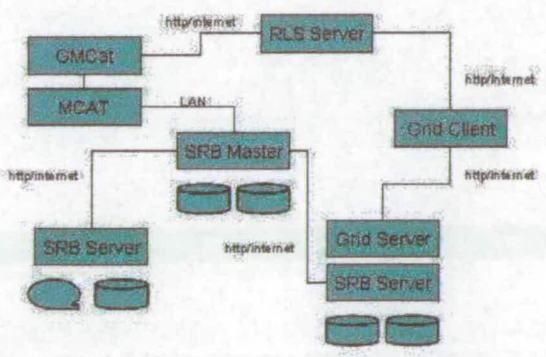

Figure 6.3: The GMCAT system

Grid clients are then able to request data directly using a Grid Interface, or other transfer mechanism such as *ftp* and *scp*, from the host. There are several issues in this that become apparent when the existing software is analysed. These are centred on the security implications of the storage mechanism SRB uses and the current method of local authentication with Grid interfaces. They show that implementing the system in this way would be unacceptable both to the local system administrators and to the users.

SRB systems can replicate data to multiple locations and use a Vault to store these files. The Vault is a directory controlled by the account which runs the SRB server (typically *srb* or *srbadmin*). Access to the files in this Vault are based on the Access Control Lists (ACLs) stored in the MCAT and SRB authentication, but are not typically encrypted on the resource. The ACL is not stored on the local resource, and authentication is performed at the SRM Master using the MCAT not on the local resource.

In practise, this means that the account running the SRB server has direct access to all the files in the local Vault and can both read and write to them. SRB maintains a hash value or checksum for each file in the MCAT. Any change to the file results in a change to the hash value and a warning to the user when the file is accessed. For the Grid interface to provide access to these files we would require the *grid-mapfile*, which provides the mapping between Grid Certificates and local accounts, to map all users to the SRB account.

This is impractical for several reason. It would bypass the SRB ACLs, and it does not scale well, as multiple clients using the same account can create conflicts. It would be very difficult to create a log of the commands each user executed if there were several users using the same account at the same time. This lack of restrictions and control would be unacceptable to both users and administrators.

The loss of security may not be a significant concern for groups who are wanting to publish data, if it is possible to ensure that this is in a read-only format. This could be achieved by installing a Grid interface at each SRB server which maps users to a profile in the same group as the *srbadmin* account, but which does not have write permission. The access control lists could be published in the same way as GMCat publishes file locations, to a VOMS which updates sites on a daily basis.

However, the scalability issues with this design are significant. Each site in the SRB system would be expected to install a Grid interface, with the necessity of various Grid Certificates, and would need to install a method of updating their ACLs regularly. While this design may work, there are other, better, solutions.

A Tape System Based Design

From the conclusions in the last section, and our knowledge of the Berkeley SRM, we investigated the possibility of treating SRB as a tape system. In Figure 6.5 we present a diagram of the Berkeley SRM. This has been designed for disk and hybrid systems which have both disk and tape. In these systems the software is designed to copy the data from tape onto disk and then transfer it across the network to the client. The disk is regarded as a cache where files are volatile, and can be deleted by the system when space is required.

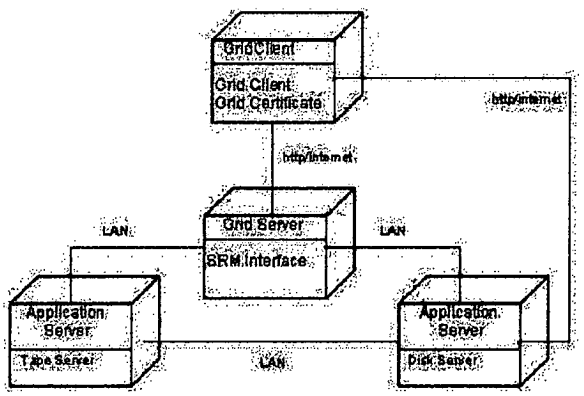

Figure 6.4: Deployment Diagram of the Berkeley SRM

We show in Figure 6.5 how the system would appear. A single SRM2SRB interface would be required at the SRB Master. Files would be replicated from whichever SRB server they are resident on to a disk cache, and then copied or transfered to the client.

The problems with this design are that it is inefficient if files reside on the local disk as they must be replicated locally before being transfered. It would only work with smaller files, since large (1TB+) files may not be able to fit in the disk cache. Remote files would need to be copied to the local system before being copied to the

user, resulting in two network transfers. With the current GMCat system the file location, which has been translated into RLS format, would need to be translated back to SRB format, resulting in another network query.

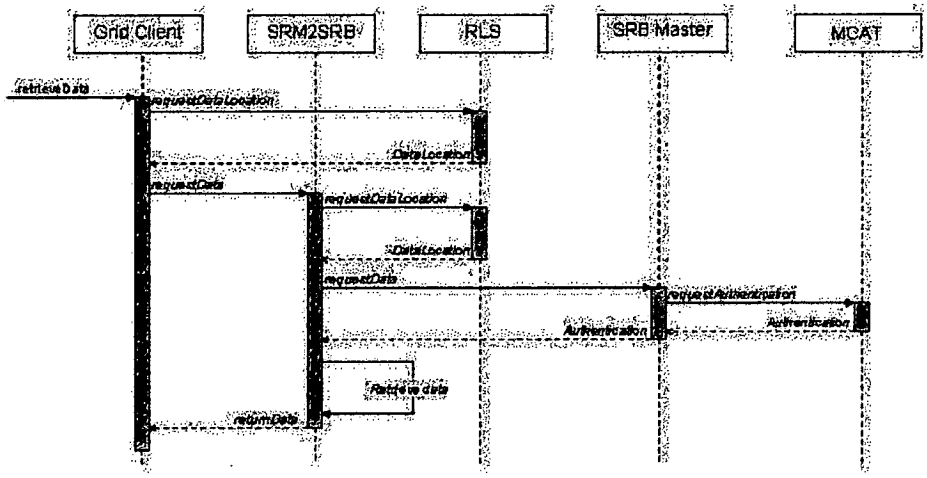

Figure 6.5: A Sequence Diagram for SRM2SRB

However, this design would have advantages. We would only need to install software on the machine hosting the master server, rather than on every node. This is obviously beneficial from a system administration and support perspective. Also, assuming the master node has a large proportion of the data and resources, which a cursory examination of the SDSC SRB system suggests is likely, the performance of the system would then be more limited by disk performance than the network.

SRB Client Design

The final design we propose for the SRM2SRB system treats SRB as a centralised resource. In this model, Figure 6.6, we make use of the SRB design, which we discussed in Section 6.3.1. Instead of treating it as a tape system we accept that it is a hybrid in its own right, and we should make use of its capabilities and strengths. We can see several potential use cases for this system. These include data retrieval from the SRB Master and SRB nodes, and data addition to the SRB Master. They are presented in more detail in Appendix C.

The main issue we faced with the direct access design was the lack of security. If we consider SRB as a system, rather than just the files it manages, then it is logical

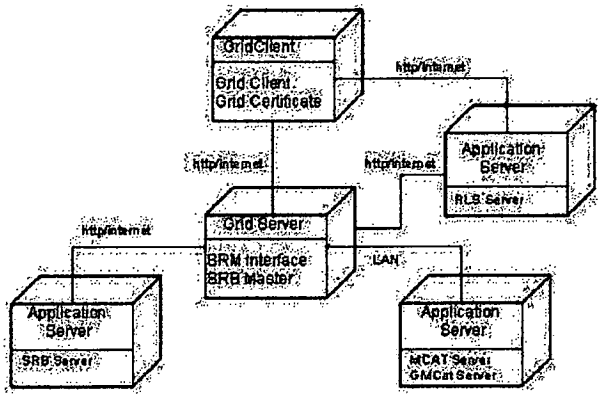

Figure 6.6: A Deployment Diagram of the SRM2SRB System

to use the interfaces to SRB which are available. Assuming the users have Grid Certificates, which is a requirement for SRM, we solve the issue of access control by using the existing system rather than creating a new one. This not only reduces the amount of development work required, but also conforms to the improved security of software development we advocated in the conclusions to Chapter 4.

Scalability was the main issue faced with the tape system based design. Our concern about the need for a Grid interface on top of every SRB resource, and the amount of effort needed to achieve this, would be reduced to a single installation at the SRB Master. Unfortunately, we would face the same inefficiency as the tape system, with data being transfered over the network twice if it is unavailable on the local resource.

The one feature which would provide interoperability between the systems is GridFTP. The main problem we face in this area is that SRB uses a parallel file transfer system of its own. Although work has been done on developing a GridFTP media driver by the SRB team, whom we have had discussions with on this, there is, as yet, no publicly available version.

Using this design of the SRM2SRB system we intend to develop the interface using the Jargon API. This is the Java API for SRB. In the next section we provide an overview of how we intend to provide the required SRM functionality using SRB.

6.4.4 Development

During the course of designing the system we realised that development using a single programming language has advantages. These include: simplified design, as the paradigm remains constant between components; easier communication between developers, as each programming language has its own idiosyncrasies and dialect; and simpler maintenance, as maintainers only require knowledge of one language. Given the choice between C++ and Java, we chose the Java programming language. Java was chose over C++ because, although SRB has API support for both, the Berkeley SRM, which has been developed in C++, is the less documented and more complex [141] of the two. The Java-based FNAL SRM has clearer documentation, as it was designed for a file system rather than an HSM, as previously stated.

We now present an overview of the core areas of the SRM specification, and how these relate to an SRB implementation. We have used the five areas proposed by the SRM working group. These are storage management functions, permission functions, directory functions, data transfer functions, and management functions. A more detailed look at how each of the functions required by the SRM specification can be achieved using SRB, is presented in Appendix C.

Storage Management Functions

The SRM storage management functions have two roles. Their practical role is to manage the storage area, create reservations for users, release reservations as appropriate, update the management functions about the available space and outstanding reservations, and remove volatile files which exist past their lifetime. In addition to this they manage access control tokens and modify the metadata of files to move them between volatile, durable and permanent space. They provide information through metadata about the resource and the managed files.

Many of these functions are difficult to address with SRB. As SRB is a distributed system in its own right, it is not necessarily limited by the resources of the node being queried. In addition, SRB manages its own files, so all those within it can be classed as either durable or permanent in SRM nomenclature. We avoid the necessity of implementing many of the required functions with anything other than token responses if we maintain SRB as a read-only resource as discussed in Section 6.4.2.

If we wished to develop the system past this point, we would need to accept that, given the current design of the SRB system, we would not be able to achieve full compliance with the SRM specification. We note that at present no other group has achieved full SRM Version 2 compatibility.

Permission Functions

The functions related to file permissions include setting permissions on new files, reassigning files to users and checking the existing permissions on a file or directory. All these can be achieved with SRB although we may not wish to implement the first two if the system is read-only. We will require the functionality necessary to check if a user has access privileges to an existing file, as this is not addressed by GMCat, and would be a necessary part of using the SRB interface.

Directory Functions

The SRM directory functions are intended to provide the methods necessary to create, list, modify, move and delete directories and to move and remove files. With an initial version of SRM2SRB we do not need to provide any of this functions. As users will only be allowed to read data from the system, they will not require the ability to create, modify, move or delete directories. Nor will they need to move or remove files within the SRB system.

The GMCat project has removed the need for us to list the contents of a directory or SRB collection, as users are given the single option of requesting files individually. However, as the software matures we would envisage that we would map the functionality in the section to Collections - the abstract directory structure of SRB - as explained in Appendix C.

Data Transfer Functions

The data transfer functions required by SRM are the ones of most interest to this project. They include protocol negotiation, which we intend to limit to GridFTP initially; get and put methods which include 3rd party transfers; multiple file transfer, which the current information system will not support; and the pinning of files.

Additional functions include the ability to request an identifier for a job, the status of a job, job suspension e.g. in the event of a server failure, resumption of a job, lifetime extension for a volatile file if the data transfer is slower than expected and job cancellation. These can all be achieved with the SRB API.

Management Functions

The management functions of SRM are often the ones overlooked or unimplemented, as they do not provide the core functionality the users request, but rather, the functionality system administrators need.

These functions include namespace management, the accounting of user requests, files access information and audit of modifications made to the system as well as the ACLs for files and directories. In our situation, issues such as accounting, audit and ACLs are provided by the SRB API. Unfortunately, namespace management becomes a more complex issue as RLS and SRB use different systems.

The method GMCat uses to address this, mapping one onto the other, is not necessarily ideal for our situation. If we implement the SRM2SRB system as proposed, it may be more logical to register data in the RLS using a different scheme. This would set the access protocol to SRB, the server for all data, as the SRB Master and the path as the MCAT stored file location information.

6.5 Conclusions

The initial work we have conducted in providing an analysis and design for an SRM interface to SRB can be used to assist in the development of an SRM interface to the University's Storage Area Network. The knowledge we developed while looking at the potential ways to create this interface, can be reused when investigating the addition of other mature data management products to the Grid.

The work we did in training an MSc student, in the deployment of the various packages, to a point where he could become a useful developer was instructive. Data management in a distributed environment is a difficult area of research even when using only one solution. When developing software for two systems simultaneously these issues are magnified. While it is possible for a single person to develop such

a solution, we believe that a greater manpower effort would improve the chances of success.

It is Edinburgh's intention to continue the development of this interface and to test this in an active environment. ScotGrid, and the GridPP2 storage RA position, intend to extend the work we have conducted so far. This should take advantage of the institutional knowledge we developed through our work with the Storage Resource Broker team at SDSC and the Storage Resource Manager teams at LBNL and FNAL.

Chapter 7

General Conclusions

Since the start of the e-Science programme in 2001 the United Kingdom's institutional knowledge of e-Science, the Grid and related areas has grown vastly. This thesis has addressed some of the issues high energy experimental particle physics will face in moving active experiments to the Grid, and how this knowledge can be used to benefit future experiments. Specifically, we have looked at the role of Tier 2 centres within the emerging infrastructure, and how these can be developed and supported.

At the end of each chapter we presented the conclusions which could be drawn from that specific section of our work. We now present some general conclusions from our research. These are intended to incorporate the knowledge and experience from all the work we have conducted, but are conclusions that do not fit within the context of a specific chapter.

We reach a general conclusion about software deployment from our experiences with the BaBar and LHCb experiments, and with the Globus, EDG and LCG software. Lightweight and self-contained solutions are the only way to get the user community to make use of the software that is being provided. The manpower available for administration at Tier 2 and Tier 3 sites is unlikely to increase prior to the start of LHC and may be limited in experience. There are two clear solutions to this issue: maintain all the complexity at the Tier 1 sites, or provide greater documentation and support for installation at the Tier 2 sites.

From the work we have done in the area of data storage we can see a clear trend

away from development at the hardware layer, and more interest in the middleware. This is a reaction to the changing nature of applications, and the move toward distributed systems as network bandwidth becomes cheaper. At present much of this change is being driven by the user community, and not always toward viable long term solutions. The data management community needs to start taking the initiative, and directing the longer term aims of this work so that scalability and security issues are addressed now.

The debate about the role of Tier 2 centres, and their relationship to their local institutions and the national and international laboratories with which they collaborate, is unlikely to be resolved easily. From our experiences with ScotGrid we can see that the policies and demands of the local institution, such as firewalls and effort allocation limit, what is achievable to a greater degree than demands from the laboratories. This can clearly be seen in Section 5.2.4 where the policies of the local Durham site were considered to outweigh the needs of the ScotGrid resources.

This feeds into the significant question of whether Tier 2 centres are a viable, long term part of the LHC computing model, or like the Tier B centres for the BaBar experiment, a stop-gap measure. With BaBar, the Tier B sites were used to gain experience and, if critical mass could be reached, were expected to become Tier A sites or revert back to Tier C status.

There are significant differences between the political and technological situations in which the BaBar and LHC experiments have been developing their respective computing models in. During the creation of the first BaBar Computing Model network bandwidth was less available than it now is, and processing power was more expensive. Resources were allocated specifically to a discipline or project. We are now reaching a point where computing has become a utility, and systems can be purchased to support multiple projects.

From the outset, the LHC computing model assumed that the computational resources available at the Tier 2 sites would also be used by earth sciences, bioinformatics and other disciplines. In the case of ScotGrid, and the other Tier 2 sites in the UK, this has proved to be correct. This has benefits - administration and manpower costs are shared between multiple groups, and the aggregate resources are greater than could be acquired individually. The problem with this approach is the

lack of consistency between disciplines, and even groups within the same discipline, in terms of their legacy infrastructure.

At present there are a cacophony of different operating systems, compilers, interfaces, programming languages and environments. The current solution to this is to provide information on the available resources and software and use resource brokers to match these to requests. This takes advantage of the common infrastructure, such as the certificate authority and VOMS, for multiple disciplines. This solution does not necessarily provide access to a wider range of resources than would otherwise be available, but it should make more efficient use of them.

Another approach, which we believe has much greater long term potential, is to provide an abstract layer or sandbox on the resources. This would address the concern administrators currently have with remote and often unknown users accessing their resources with limited restriction. Providing this level of security may persuade more institutions to make their resources available on the Grid, and would simplify programming for this environment. We therefore believe that either homogeneity of resources, or better information about resources, in terms of software capabilities, lie at the heart of making the Global Grid a reality.

Appendix A

Grid Interface Performance

A.1 Introduction

The performance of Grid resource managers and their scalability is interesting as it has a direct bearing on the number of host machines needed by an experiment. Ensuring that we have a sufficient minimum means that we can provide an acceptable level of service without having idle resources. The time taken for installation, complications that occur, security and concurrent users issues are also of interest as they represent the cost, in system administrator time, which the system will require.

We initially looked at Version 2 of the Globus Toolkit as a portal into resources at SLAC and ScotGrid. This was the most widely available distribution at the time (2002/3) and was used as a basis for development work by the European DataGrid project. Version 2 of the Globus Toolkit went through eight publicly available iterations. We tested two of these, 2.1.0, the first stable release, and 2.4.3, the final 2.x release.

A.2 Experimental setup

Tests were conducted on a IBM x305 dual PIII-1GHz with 2GB of memory and 16GB of local storage running RedHat 7.3 on the 2.4.20-18.7smp kernel. System load was minimal at the start of each test but as this is not a batch system it is possible that processes owned by other users and the operating system may have used some processing time. Times were taken using the GNU *time* utility which provides user, system and wall clock times. The user and system results from the

time utility measures elapsed time for the program and operating system calls in the CPU core. It should therefore be unaffected by other users of the system. This does not apply to the wall clock time which will be discussed later.

Installation of the Globus software was relatively straight forward although some issues caused problems. Some of the areas where we found difficulties included ensuring that the system was registered for reverse DNS lookup, that the host certificates were in the correct directory, that the logical or soft links to the relevant configuration files were in place, and that site specific firewall issues were addressed.

These issues were significant at the start of our investigation (2002) as there was a limited general knowledge of the Globus software. We discovered that both DNS and reverse DNS are used to authenticate host machines. Our machine had initially only been configured for DNS registration. Site firewall issues were also a common problem during the early releases of the Globus Toolkit. This was later solved with the *GLOBUS_TCP_PORT_RANGE* environmental variable, which allowed system administrators to open specific ports for client requests.

We created a local certificate authority using the Globus SimpleCA package. This was used to provide a host certificate for the machine and user certificates for the 20 client accounts we created. Each of these certificates was based on a 1024 bit key. The client accounts were given space in the */home* directory and the ability to log into the system directly as well as being mapped to.

A.3 The effects of the proxy key length

The length of the encryption key used in the users proxy certificate will have an effect of the time taken to perform the mutual authentication handshake between the client and the server. In this test we are interested to discover what effect this has on performance; does an increase in key size lead to a logarithmic, linear, or exponential increase in the time taken to perform the same basic request?

We tested this by creating a single client - located on the same machine as the gatekeeper to remove potential network latency - which submitted ten sets of 150 requests using each of the permitted key lengths in Globus 2.x. We performed this test on a permanently available gatekeeper run using Init and then re-ran the test

using a gatekeeper which was dynamically created using the Xinetd process.

Time	Key	Low	Mean	High	Std. deviation
User	512	0.090	0.171	0.270	0.026
	1024	0.220	0.296	0.400	0.027
	2048	0.930	1.050	1.120	0.026
System	512	0.050	0.129	0.270	0.031
	1024	0.060	0.129	0.250	0.029
	2048	0.060	0.136	0.270	0.032
Real	512	6.598	6.992	7.992	0.032
	1024	6.923	7.704	9.992	0.566
	2048	7.944	13.91	42.925	4.095

Table A.1: Simple request with varying proxy key lengths using an Init 2.1.0 gatekeeper in seconds

Time	Key	Low	Mean	High	Std. deviation
User	512	0.090	0.172	0.260	0.026
	1024	0.210	0.296	0.380	0.025
	2048	0.940	1.046	1.120	0.026
System	512	0.040	0.131	0.240	0.029
	1024	0.050	0.133	0.270	0.032
	2048	0.060	0.137	0.350	0.037
Real	512	6.787	6.996	8.052	0.067
	1024	6.909	7.706	9.992	0.552
	2048	7.989	14.23	33.995	4.346

Table A.2: Simple request with varying proxy key lengths using an Xinetd 2.1.0 gatekeeper in seconds

A.3.1 Proxy Key Length Conclusions

As we can see from the summary of results in Table A.5 and in Figures A.1 and A.3, as we increase the length of the proxy being used, we find a corresponding exponential increase in the *user time* take by the system to process each request. It is worth noting that the standard deviation remains relatively constant, both with the software releases and invocation method.

The results of the *real time* taken by the system to deal with requests, summarised in Table A.6 and Figures A.3 and A.2, shows that the performance of the software has improved significantly between the first and final release - due mainly to the

Time	Key	Low	Mean	High	Std. deviation
User	512	0.130	0.223	0.320	0.029
	1024	0.260	0.357	0.480	0.029
	2048	1.010	1.136	1.230	0.030
	4096	6.010	6.150	6.240	0.032
System	512	0.040	0.125	0.210	0.025
	1024	0.060	0.131	0.280	0.029
	2048	0.050	0.132	0.260	0.030
	4096	0.050	0.138	0.280	0.031
Real	512	1.008	1.129	1.928	0.093
	1024	1.124	1.263	2.392	0.131
	2048	1.802	1.909	2.713	0.082
	4096	6.863	6.982	7.960	0.086

Table A.3: Simple request with varying proxy key lengths using an Init 2.4.3 gatekeeper in seconds

Time	key	low	mean	high	Std. deviation
User	512	0.120	0.223	0.320	0.030
	1024	0.270	0.354	0.450	0.029
	2048	1.030	1.135	1.240	0.028
	4096	6.040	6.157	6.870	0.054
System	512	0.060	0.131	0.260	0.028
	1024	0.060	0.125	0.220	0.027
	2048	0.050	0.132	0.250	0.027
	4096	0.050	0.147	0.790	0.063
Real	512	1.009	1.165	2.266	0.139
	1024	1.118	1.257	2.205	0.124
	2048	1.813	1.928	2.774	0.109
	4096	6.876	7.041	10.603	0.264

Table A.4: Simple request with varying proxy key lengths using an Xinetd GT 2.4.3 gatekeeper in seconds

		2.1.0 Gatekeeper				2.4.3 Gatekeeper			
	Key length	512	1024	2048	4096	512	1024	2048	4096
Init	Mean	0.171	0.296	1.05	-	0.223	0.357	1.136	6.150
	Std. deviation	0.026	0.027	0.026	-	0.029	0.029	0.030	0.032
Xinetd	Mean	0.172	0.296	1.046	-	0.223	0.354	1.135	6.157
	Std. deviation	0.026	0.025	0.026	-	0.030	0.028	0.028	0.054

Table A.5: A summary of user time results from varying proxy key lengths using a simple request in seconds

		2.1.0 Gatekeeper				2.4.3 Gatekeeper			
	Key length	512	1024	2048	4096	512	1024	2048	4096
Init	Mean	6.992	7.704	13.909	-	1.129	1.263	1.909	6.982
	Std. deviation	0.032	0.566	4.095	-	0.093	0.131	0.082	0.086
Xinetd	Mean	6.996	7.706	14.231	-	1.165	1.257	1.928	7.041
	Std. deviation	0.067	0.551	4.346	-	0.139	0.124	0.109	0.264

Table A.6: A summary of real time results from varying proxy key lengths using a simple request in seconds

involvement of algorithm experts and the resulting improvements in the code. We knew already that the real time would be significantly higher than user time, as it includes the time taken for communication and the servers processing time for both performing the authentication and authorisation and dealing with the request.

The effect of the length of the key can also be clearly seen in the exponential curves of graphs A.1, A.2, A.3 and A.4. This is a result of the exponential increase in the complexity of the key as its length is increased e.g. for each extra bit the key length is increased the domain of potential values doubles. From this it is obvious that although increasing the key length will theoretically make our communications more secure there will be a penalty in processing overhead.

A.4 Multiple Globus clients

It is unlikely that there will be a separate gatekeeper system for each user as this would negate the advantages of the Grid model. It is therefore, of interest to know what the effect of increasing the number of concurrent users will have on the system. We assume that at some point, as the number of clients increase, the primary gatekeeper process i.e. the gatekeeper assigned to the well known port, will become saturated with requests and start either losing requests or failing to complete them.

We used 20 basic client accounts, created with the default shell and no special environment, to test the behaviour of the system with 1, 2, 5, 10 and 20 clients submitting requests in parallel. For each test ten sets of 150 requests were submitted. As we used a human operator to start these processes there is a small delay between the first and last process being started. To remove this from the results we ignore the first set of results and use the remaining nine sets.

We selected 20 clients to be the maximum number used for our tests based on the following assumptions and back-of-envelope calculation. The BaBar experiment involves approximately 500 physicists, who live in different time zones, and are therefore unlikely to all submit requests simultaneously. Given that there are five Tier A centres, and a similar number of institution level resources we calculated : 500 physicists ÷ 2 time zones ÷ 5 centres ÷ a less than 50% chance they submit a request. This suggests that an approximation of 20 simultaneous users for a resource is an acceptable number for testing.

	Clients	Low	Mean	High	Std. deviation
User	1	0.090	0.171	0.270	0.026
	2	0.090	0.175	0.260	0.027
	5	0.070	0.164	0.270	0.026
	10	0.070	0.168	0.280	0.026
	20	0.080	0.172	0.290	0.027
System	1	0.050	0.130	0.270	0.031
	2	0.040	0.130	0.250	0.030
	5	0.020	0.108	0.260	0.029
	10	0.030	0.115	0.270	0.030
	20	0.020	0.126	0.330	0.033
Real	1	6.889	6.992	7.992	0.026
	2	6.689	6.994	7.997	0.052
	5	6.840	9.132	20.989	2.661
	10	6.938	8.820	11.103	0.468
	20	6.961	12.502	17.942	1.116

Table A.7: Results from varying number of concurrent users using an Init GT 2.1.0 gatekeeper for a simple request in seconds

	Clients	Low	Mean	High	Std. deviation
User	1	0.100	0.173	0.260	0.025
	2	0.090	0.176	0.280	0.027
	5	0.080	0.161	0.260	0.025
	10	0.060	0.168	0.270	0.026
	20	0.080	0.171	0.290	0.027
System	1	0.050	0.132	0.280	0.033
	2	0.050	0.134	0.270	0.031
	5	0.030	0.105	0.270	0.029
	10	0.020	0.117	0.300	0.033
	20	0.020	0.124	0.330	0.034
Real	1	6.531	6.995	7.992	0.056
	2	6.545	7.0	7.998	0.098
	5	6.870	7.874	9.100	0.347
	10	6.975	8.946	11.136	0.504
	20	6.952	14.647	19.236	1.082

Table A.8: Results from varying number of concurrent users using an Xinetd GT 2.1.0 gatekeeper for a simple request in seconds

	Clients	Low	Mean	High	Std. deviation
User	1	0.130	0.220	0.330	0.028
	2	0.110	0.222	0.330	0.030
	5	0.120	0.221	0.340	0.030
	10	0.110	0.222	0.330	0.029
	20	0.020	0.225	0.340	0.030
System	1	0.050	0.128	0.210	0.025
	2	0.040	0.129	0.250	0.029
	5	0.020	0.126	0.260	0.028
	10	0.040	0.130	0.270	0.029
	20	0.100	0.139	0.340	0.032
Real	1	1.014	1.129	2.438	0.101
	2	1.167	1.493	3.279	0.211
	5	1.097	3.491	6.076	0.440
	10	2.359	7.353	11.289	1.034
	20	3.066	15.183	31.311	1.815

Table A.9: Results from varying number of concurrent users using an Init GT 2.4.3 gatekeeper for a simple request in seconds

A.4.1 Multiple Client Conclusions

Figures A.5 and A.6 present the real time results for multiple clients using the 2.1.0 and 2.4.3 toolkits with both Init and Xinetd gatekeepers. The figures show that the initial performance issues the Globus software experienced were significantly reduced by the final release. Our results show that the time take to complete a simple request, for a single client, decreased from near seven seconds to just over one.

However, in both releases the real time taken to complete a simple request with 20 simultaneous clients is very close. As we can see from the error bars in Figure A.6 the 2.4.3 release had a much greater variance in resolution times. This was due to some spurious results, including a maximum recorded resolution time of over six minutes for the Xinetd gatekeeper. This suggests that the system can become saturated with requests but will eventually honour them. This has consequences for the quality of service which are discussed in Section 4.5.1.

The results of these tests suggest that the major processing cost is with the start up of the initial gatekeeper and after this there is a smaller processing cost for each additional handler. In all four cases the increase in resolution time appears to be closer to linear than exponential. The general conclusions from this, in terms of system management, are discussed in the next section.

A.5 General Conclusions

In addition to answering our original questions the tests we conducted have lead us to some general conclusions. Our results, from investigating the proxy key length, show that the majority of processing for job submission is conducted on the user's machine. We can therefore specify a larger key size without degrading performance on the server side. The issue of slow response times will be mitigated by Moore's Law. CERN has already adopted this approach with 2048 bit keys with mixed success.

The conclusion we draw from our tests using multiple clients is that denial of service attacks are a realistic concern. The use of resource brokers as an intermediate layer between users and systems is unlikely to solve this as resource brokers will have similar issues to contend with, and will be subject to network latencies which

a resource allocation manager on a single machine will not. This remains an open question for research.

We conclude by reminding the reader that these results only apply to situations where simple requests are submitted using Version 2.x of the Globus Toolkit. More recent developments, such as using pooled accounts for job execution, resource brokering systems and data management systems have not been investigated. These areas would be of interest for further study as they would assist in the development of a Grid simulator. This could be used to aid our understanding of the Grid and assist in purchasing and management decision making.

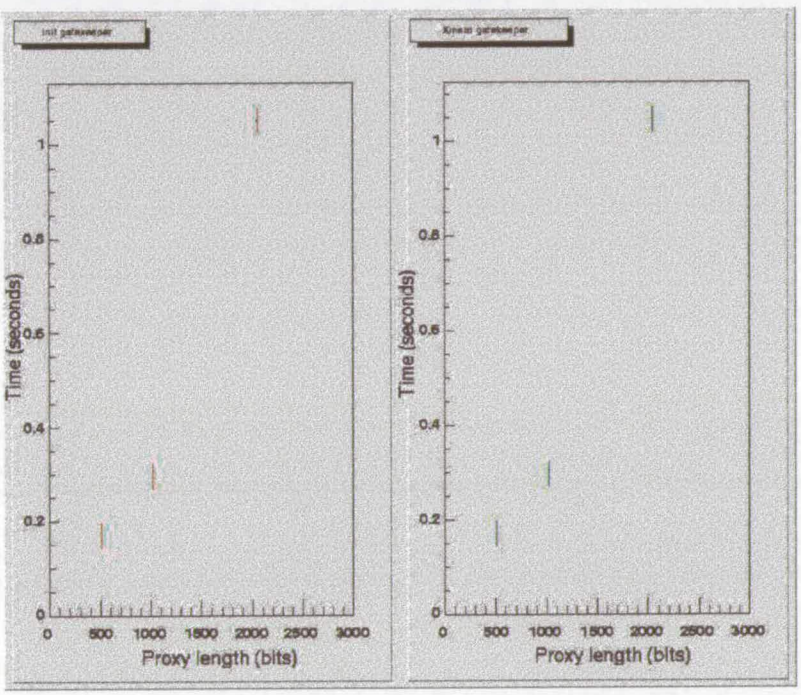

Figure A.1: Effect on user time of increasing the proxy key length with 2.1.0 gatekeeper

	Clients	Low	Mean	High	Std. deviation
User	1	0.120	0.223	0.330	0.031
	2	0.120	0.222	0.310	0.030
	5	0.120	0.224	0.350	0.030
	10	0.120	0.222	0.340	0.030
	20	0.100	0.223	0.350	0.030
System	1	0.030	0.133	0.310	0.032
	2	0.060	0.130	0.240	0.028
	5	0.030	0.135	0.290	0.032
	10	0.040	0.132	0.330	0.029
	20	0.010	0.134	0.400	0.031
Real	1	1.004	1.189	2.329	0.169
	2	0.674	1.557	3.331	0.285
	5	1.084	4.192	7.791	0.769
	10	2.300	7.882	18.194	0.960
	20	1.973	16.481	363.271	6.518

Table A.10: Results from varying number of concurrent users using an Xinetd GT 2.4.3 gatekeeper for a simple request in seconds

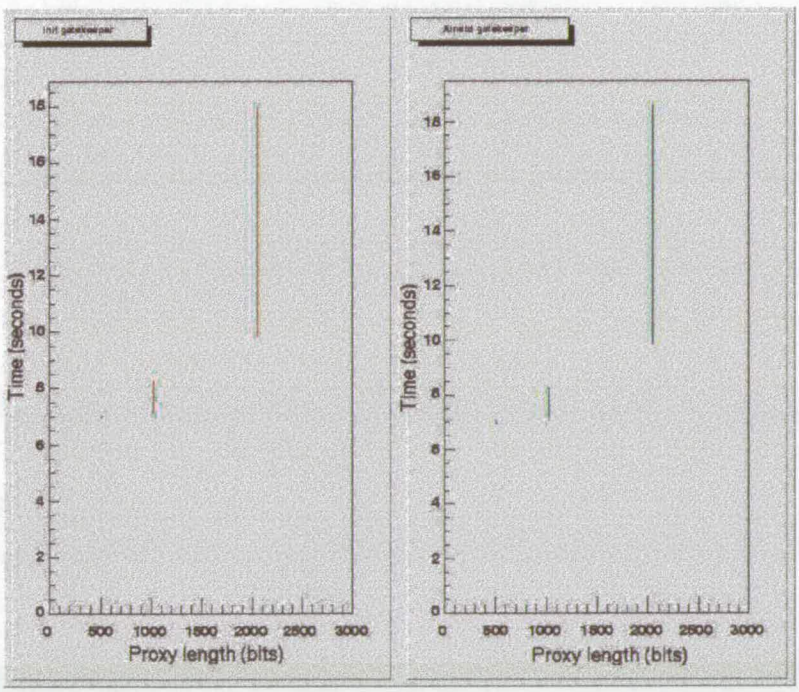

Figure A.2: Effect on real time of increasing the proxy key length using a 2.1.0 gatekeeper

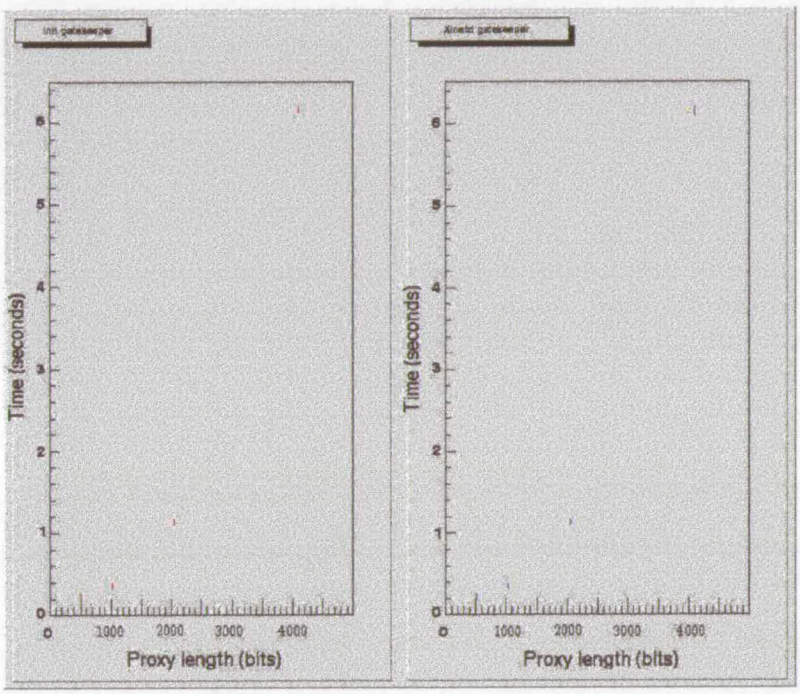

Figure A.3: Effects on user time of increasing the proxy key length using a 2.4.3 gatekeeper

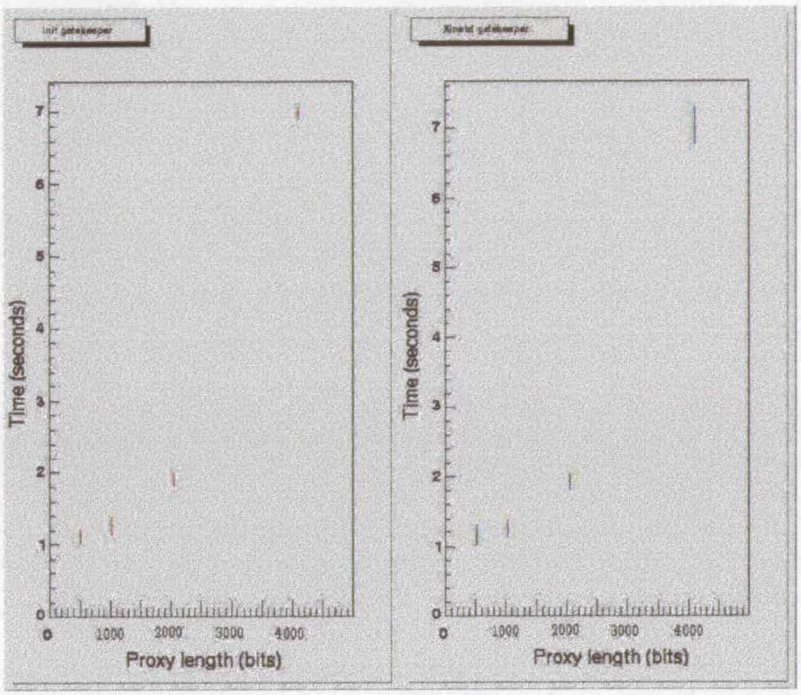

Figure A.4: Effects on real time of increasing the proxy key length using a 2.4.3 gatekeeper

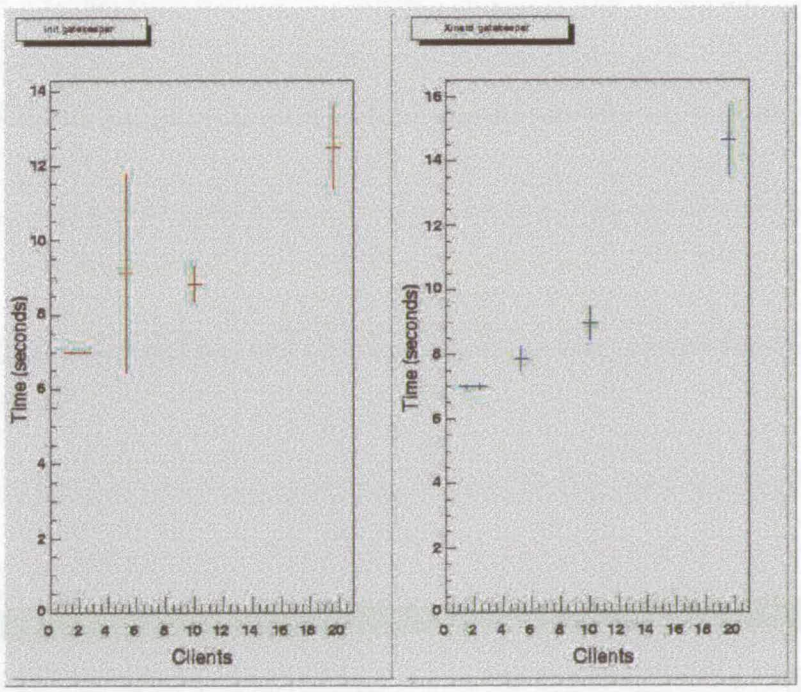

Figure A.5: Effects of increasing numbers of clients on resolution time with 210

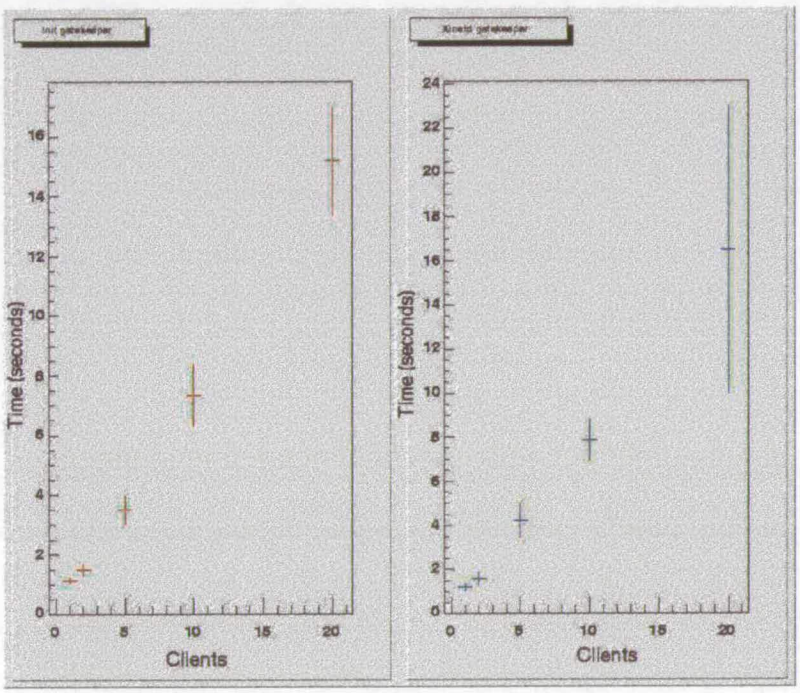

Figure A.6: Effects of increasing numbers of clients on resolution time with 243

Appendix B

High Performance Linpack on Glenkinchie

B.1 Introduction

The ScotGrid installation at Edinburgh is primarily focused on providing storage facilities to the project over its lifetime. However, we have discovered that some computing capabilities, above the absolute minimum, are necessary for system management and to allow Edinburgh users to develop software and knowledge. This was discussed in Section 5.3.

The group currently possesses a high performance shared memory machine, Glenkinchie, which is used to manage the RAID arrays using two FAStT900 controllers. Recently, we have also taken delivery of four dual processor machines which are yet to be installed and could be used for a variety of tasks.

Although the majority of simulation production and analysis in particle physics research can be regarded as loosely coupled, some of the projects which use ScotGrid, such as QCDGrid, benefit from access to shared memory machines. As part of our continued work to make ScotGrid a useful and attractive Tier 2 centre we have benchmarked Glenkinchie to provide an indication of its potential performance.

B.2 Experimental Setup

These tests were conducted on an IBM eServer x440 (Glenkinchie) which possesses eight Intel Xeon MP 1.9GHZ processors and 32 Gigabytes of memory. It is connected, using two IBM FAStT900 RAID controllers and a fibre connection, to two arrays which have 24TB of disk. Glenkinchie also has 30GB of local disk which provides the operating system and binaries. We stress that Glenkinchie is not configured for true batch processing, as a result, performance is expected to be significantly less than the theoretical maximum.

The Intel Xeon architecture is theoretically capable of performing two floating-point additions and multiplications (in full precision) during each clock cycle. This gives us a theoretical maximum, or peak performance, of 3.8 GFlops per processor which is 30.4GFlops for the system. A cursory view of the Top500 list of supercomputers however, tends to indicate that most sites running Xeon based machines are achieving between 50 and 70% of this peak rating after optimising their systems.

B.2.1 High Performance Linpack

The High Performance Linpack (HPL) benchmark [41] has been developed to test the processing capabilities of distributed memory systems. It is currently used as the basis of the rankings for the Top500 list [78] of the world's most powerful supercomputers. Linpack uses sparse matrices calculations, which are memory resident, as the basis of its results. Using only memory for its data storage avoids the latency experienced when data is read from, or written to, disk and provides a better indication of processing capability.

Using HPL we are able to vary several factors including matrix size, block size, floating point accuracy and how messages are passed. We have used several tests based on IBM recommendations [132] for the x440 series of machines. These tests were conducted using the ATLAS libraries and then re-run using the Goto Libraries for comparison. In both cases we tested a 1 $\times$1, 2$\times$2 and 2$\times$4 processor arrangement.

ATLAS Libraries

The Automatically Tuned Linear Algebra Software Libraries (ATLAS) [4] provide the libraries necessary for performing basic linear algebra and are a superset of the

Basic Linear Algebra Software (BLAS) [11]. These, or libraries with equivalent functionality, are required when compiling the Linpack software. The version used for these tests is 3.6.0.

GOTO Libraries

Kazushige Goto of the University of Texas - Austin has created a highly optimised BLAS library which is freely available and widely used [38]. These are believed to provide a better level of performance than the ATLAS libraries. To discover if this is true we ran a variety of tests using ATLAS and then performed identical tests with HPL compiled against the GOTO libraries. The version used for these tests was compiled for the Intel P4 with 512k of L2 cache - libgoto_p4_512-r0.94.so.

B.2.2 Intended Experiments

These are initial tests to discover the basic performance and properties of Glenkinchie and not exhaustive. Using Linpack in all configurations and with all potential parameters would produce in the order of ten's of thousands of results and require a similar number of CPU hours. Glenkinchie is intended for production use, therefore running benchmarking software on it reduces the availability for researchers.

Given these limits we restrict our tests to those which will provide a guide for future work in optimising the system. Our main interests are: how performance is affected by problem size; how the data block size - the amount of data copied by each processor to its local cache - affects performance; how different libraries can effect performance; how processor utilisation changes as more processors are added; and any other experience which can be gained from this.

Parameters File

The following is the parameters file used with both libraries and is based on the parameters used by IBM for testing the xSeries of clusters. Lines 6 and 6 define the matrix size which will be used; 7 and 8 define the block sizes being used; 11-12 define how many processors are used and their arrangement.

```
HPLinpack benchmark input file
1. Innovative Computing Laboratory,
2. University of Tennessee
```

```
 3. 4CPU_HPL.out      output file name (if any)
 4. 8           device out (6=stdout,7=stderr,file)
 5. 1           # of problems sizes (N) - max 20
 6. 20000       Ns
 7. 10          # of NBs
 8. 100 200 300 400 500 600 700 800 900 1000   NBs
 9. 0           PMAP process mapping (0=Row-,1=Column-major)
10. 1           # of process grids (P x Q)
11. 2           Ps 12. 2            Qs
13. 16.0        threshold
14. 1           # of panel fact
15. 2           PFACTs (0=left, 1=Crout, 2=Right)
16. 1           # of recursive stopping criterium
17. 8           NBMINs (>= 1)
18. 1           # of panels in recursion
19. 2           NDIVs
20. 1           # of recursive panel fact.
21. 0           RFACTs (0=left, 1=Crout, 2=Right)
22. 1           # of broadcast
23. 1           BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
24. 1           # of lookahead depth
25. 1           DEPTHs (>=0)
26. 0           SWAP (0=bin-exch,1=long,2=mix)
27. 64          swapping threshold
28. 0           L1 in (0=transposed,1=no-transposed) form
29. 0           U  in (0=transposed,1=no-transposed) form
30. 1           Equilibration (0=no,1=yes)
31. 8           memory alignment in double (> 0)
```

B.3 Results

The results of the tests conducted are presented in the following Tables.

	Block Size				
Matrix Size	100	200	300	400	500
100	0.013	0.430	0.438	0.447	0.448
250	0.737	0.836	0.644	0.771	0.772
500	1.420	1.357	1.216	1.047	0.961
750	1.588	1.615	1.608	1.458	1.255
1000	1.875	1.866	1.837	1.718	1.575
2500	1.686	1.913	2.029	2.061	2.039
5000	1.462	1.844	2.020	2.101	2.142
7500	1.467	1.888	2.075	2.174	2.224
10000	1.464	1.876	2.110	2.216	2.264
12500	1.491	1.894	2.123	2.187	2.272
15000	1.493	1.953	2.088	2.255	2.301

Table B.1: ATLAS results using 2 processors in GFlops

	Block Size				
Matrix Size	100	200	300	400	500
100	0.061	0.071	0.074	0.075	0.075
250	0.387	0.215	0.185	0.196	0.198
500	1.113	0.728	0.560	0.450	0.407
750	1.840	1.276	1.020	0.809	0.714
1000	2.457	2.000	1.531	1.301	0.941
2500	4.154	3.821	3.594	3.234	3.028
5000	4.224	4.588	4.686	4.690	4.212
7500	4.406	5.161	5.257	5.241	5.116
10000	4.502	5.335	5.720	5.703	5.647
12500	4.736	5.683	6.205	6.549	6.529
15000	4.477	5.703	6.112	6.312	6.166
17500	4.827	6.016	6.531	6.716	6.783
20000	4.935	6.368	6.790	6.889	6.900
22500	5.196	6.608	7.085	6.818	7.081
25000	4.898	6.164	7.032	7.378	7.423
27500	4.628	6.190	6.623	7.422	7.213
30000	5.463	6.736	7.401	7.548	7.490

Table B.2: ATLAS results using 4 processors in GFlops

	Block Size				
Matrix Size	100	200	300	400	500
100	0.023	0.031	0.066	0.024	0.082
250	0.229	0.214	0.197	0.206	0.197
500	1.002	0.628	0.501	0.368	0.388
750	1.318	1.151	1.025	0.843	0.736
1000	2.722	1.870	1.419	1.149	0.788
2500	5.912	1.537	4.229	1.558	1.903
5000	7.053	6.865	6.994	4.458	5.766
7500	7.156	7.552	8.259	7.447	6.401
10000	7.475	8.289	9.014	8.894	8.192
12500	7.158	8.885	9.189	9.458	9.187
15000	5.970	7.755	9.328	9.707	9.619
17500	6.738	8.801	9.865	9.999	9.548
20000	6.781	9.383	8.243	10.410	8.825
22500	6.843	7.859	9.214	10.790	10.770
25000	6.452	9.310	10.300	9.889	10.790
27500	6.907	9.648	10.27	12.65	12.74
30000	6.212	8.661	10.560	11.040	9.643
35000	6.503	8.825	10.790	11.110	10.850
40000	5.907	9.290	10.330	10.770	11.280
45000	5.830	9.625	10.740	10.720	10.920

Table B.3: ATLAS results using 8 processors in GFlops

Block Size	Result	Block Size	Result
20	3.15	475	8.67
25	3.27	500	8.192
40	5.24	525	8.29
50	5.85	550	9.043
60	6.47	575	8.35
75	6.92	600	8.128
80	7.31	625	7.78
100	7.475	650	8.18
125	7.2	675	7.99
150	8.77	700	7.991
175	8.21	725	8.25
200	8.289	750	8.3
225	8.11	775	7.95
250	9.15	800	7.733
275	8.55	825	7.58
300	9.014	850	8.12
325	8.85	875	7.65
350	9.37	900	7.442
375	9.01	925	7.76
400	8.894	950	8.31
425	8.68	975	7.38
450	9.25	1000	6.831

Table B.4: Results for matrix size 10,000 using 8 processors and ATLAS in GFlops

	Block Size				
Matrix Size	100	200	300	400	500
100	0.362	0.659	0.661	0.661	0.657
250	1.259	1.083	0.983	1.049	1.045
500	1.865	1.705	1.565	1.385	1.138
750	2.111	2.050	1.914	1.763	1.669
1000	2.233	2.197	2.098	2.005	1.855
2500	1.923	1.915	1.909	1.948	1.952
5000	1.919	2.011	2.010	2.037	2.036
7500	2.237	2.288	2.282	2.323	2.323
10000	2.278	2.327	2.152	2.328	2.063
12500	2.002	2.044	2.010	2.029	2.021
15000	1.990	2.024	2.020	1.954	1.817

Table B.5: Goto results using 2 processors in GFlops

B.4 Conclusions

The results of these tests clearly answer the questions we originally asked. They have also presented several areas for future work which are discussed in Chapter 7. In this section we deal with our original questions individually. Comments on the more general conclusions which can be reached from this data are discussed in Section 5.2.2.

B.4.1 Matrix Size

The results show that for both the ATLAS and Goto libraries the performance of the system typically increases as the matrix size increases. Figures B.1 and B.2 demonstrate this using the ATLAS and Goto libraries in an eight processor configuration. In both figures we can see a large performance gain as the matrix size increases past 1,000 and the performance of the system increases from around two GFlops to around 10 GFlops per second.

The low performance we see with small matrices can be explained by the overhead the system experiences in terms of inter-processor communication. With small matrix sizes more processing time must be devoted to maintain consistency between the processor caches and main memory as the relative size differences are small.

The increase in performance, seen between matrix sizes 1,000 and 10,000, we explain as being due to an increase in the difference between cache size and matrix size. Processors are able to work on subsets of the matrix without other processors modifying the same data cells. However, after this point the system does not see any significant further benefit from increasing the problem size.

From the results we obtained, increasing the matrix size from 10,000 to 45,000 does not significantly improve the performance of the system. However, the time taken to perform the calculation, rises from around 90 seconds to two hours.

B.4.2 Block Size

Linpack allows benchmarkers to vary the block size used during the tests. The block size specifies the granularity of the calculation, as well as the quantity of data which is transfered during each memory read and write. A small block size increases the

load balance, but, reduces the benefits of data reuse and increases the communication overhead. A large block size, in comparison, will allow a processor to reuse much of the data but will suffer from memory inconsistency if two processors try changing the same data block.

We chose to use the ATLAS libraries and a matrix size of 10,000 to perform our test. This decision was based on the earlier results, and the knowledge that we could achieve close to maximum performance with this matrix size without requiring excessive processing times. We used block sizes between 25 and 1000, in increments of 25, and between 20 and 100 in increments of 20. These results are shown in Table B.4.

As we can see, in Figure B.3, there is a rapid performance increase between the initial block size of 20 and a block size of 350. After this point however, the system experiences diminishing returns and the results achieved with block sizes greater than 350 show a downward trend. This is consistent with expectations and confirms the requirement for block sizes to be consistent with problem size.

B.4.3 Processors and Performance

We expressed an interested in discovering the effect of increasing the number of processors used to perform a calculation. Theory claims that there will be diminishing returns as the communication overhead increases at a greater rate than the number of processors added. Figure B.4 shows the results achieve using two, four and eight processors running Linpack with the ATLAS libraries and a block size of 300.

Initially, the system using the fewest number of processors performs the best, 0.4 GFlops vs. 0.1 GFlops. As the problem size grows, the system with the largest number becomes the best performing in absolute terms. The maximum performances achieve were 2.1 (matrix size 7500+), 7.4 (30,000+) and 10.8 (35,000) GFlops. However, when we look at individual processor performance a more interesting situation presents itself.

The maximum performance achieved per processor, based on the results we recorded was, 1.55, 1.85 and 1.35 GFlops for the two, four and eight processor configuration. This shows that there is an optimal number of processors for systems, and that

this lies at a point between the maximum and minimum. Of more interest are the results for the matrix size of 15,000, where the two processor configuration achieved its maximum performance. At this point the four processor configuration achieved only 1.525 Gflops per processor and the eight processor configuration achieved 1.16 Gflops. This clearly shows that there is a diminishing return to adding more processors while there is an absolute increase in performance.

B.4.4 ATLAS vs Goto

From the results we achieved using eight processors and a block size of 300, shown in Figure B.5, we can see that the Goto libraries have a slightly better performance for small matrices. The performance of the Goto libraries is however, overtaken at a matrix of 7500 by the ATLAS ones. At the maximum matrix size for this configuration, 45,000, the ATLAS libraries produce a 30% greater performance than the others.

This does not match our expectations. The disparity can be explained by the fact that we are using Goto libraries designed for Intel Pentium 4 processors with 512k of L2 cache rather than ones for the Xeon MP processors being used. We were unable to locate libraries designed for our architecture and we should therefore consider using other versions of the Goto libraries to discover if a different version would provide superior performance.

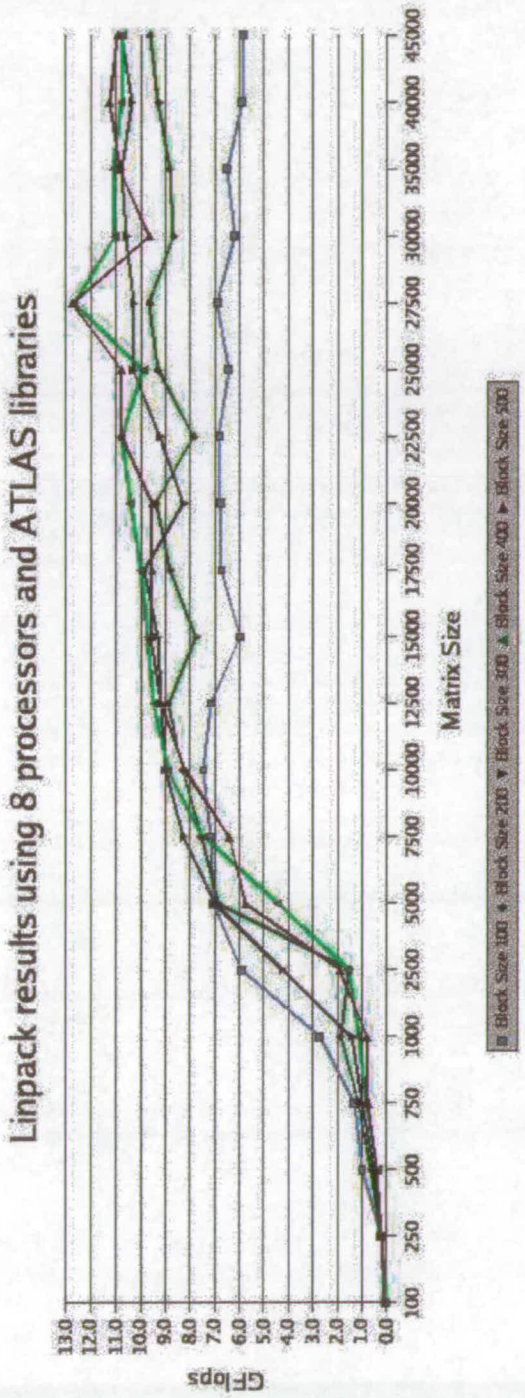

Figure B.1: 8 processor ATLAS results

Figure B.2: 8 processor Goto results

Figure B.3: Results of varying block size using ATLAS

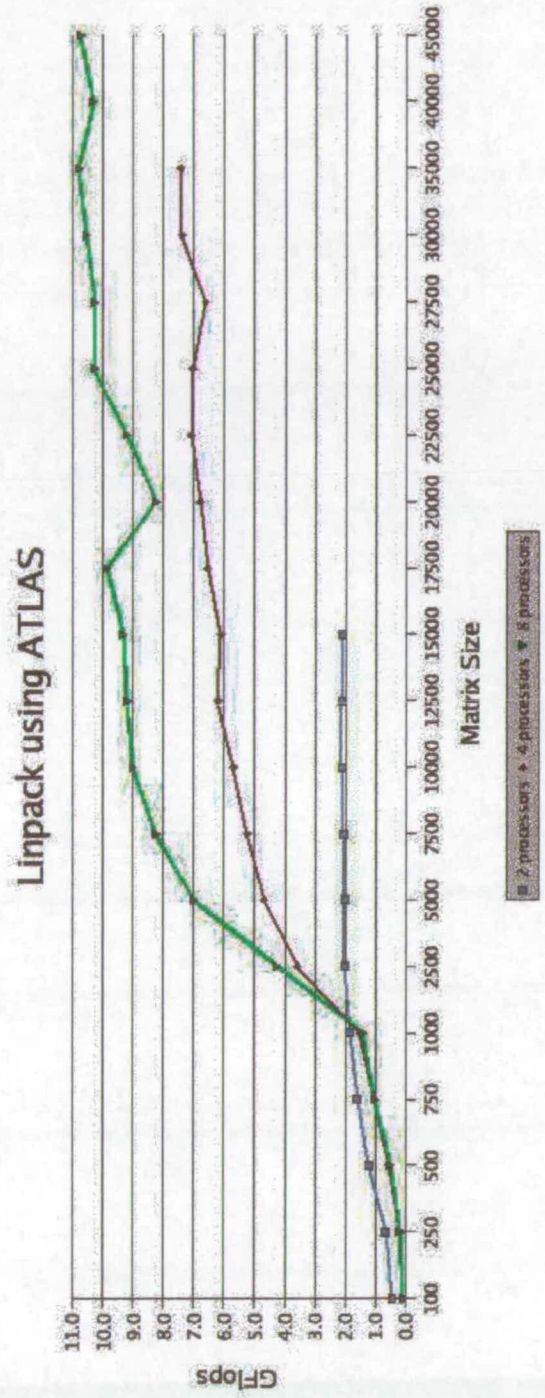

Figure B.4: Results of varying processor numbers using ATLAS

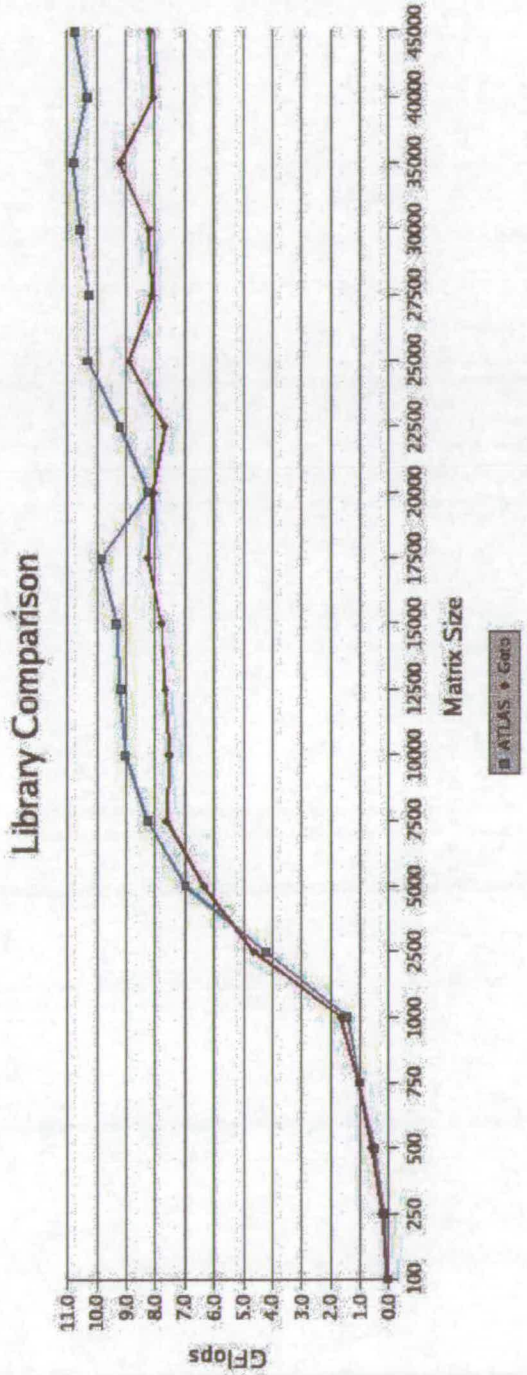

Figure B.5: Results of varying BLAS library with 8 processors and block size 300

	Block Size				
Matrix Size	100	200	300	400	500
100	0.035	0.128	0.129	0.128	0.126
250	0.391	0.343	0.363	0.323	0.323
500	1.272	0.902	0.606	0.633	0.588
750	2.123	1.561	1.255	0.912	0.900
1000	2.633	2.479	1.843	1.539	1.033
2500	4.112	4.185	3.937	3.631	3.395
5000	4.835	4.476	4.264	4.390	4.151
7500	4.606	4.506	4.891	4.890	4.842
10000	5.029	5.018	4.869	5.371	4.448
12500	5.162	5.206	5.173	5.311	5.051
15000	5.453	5.255	5.592	5.243	5.616
17500	5.730	5.735	5.726	5.789	5.089
20000	5.785	5.799	5.130	5.979	5.705
22500	4.899	5.295	5.175	5.544	5.034
25000	5.263	5.071	5.351	5.377	4.926
27500	5.190	5.136	5.288	6.107	6.054
30000	5.229	5.258	5.414	5.987	5.491

Table B.6: Goto results using 4 processors in GFlops

	Block Size				
Matrix Size	100	200	300	400	500
100	0.023	0.084	0.082	0.089	0.101
250	0.319	0.298	0.249	0.256	0.259
500	1.092	0.715	0.609	0.488	0.454
750	2.062	1.307	1.052	0.580	0.701
1000	2.871	2.283	1.660	1.452	0.982
2500	6.494	5.738	4.655	4.130	3.420
5000	7.328	7.150	6.362	6.262	5.166
7500	7.628	7.510	7.617	7.299	6.884
10000	7.643	7.564	7.522	7.967	7.013
12500	7.151	7.714	7.663	7.427	7.495
15000	7.495	7.549	7.791	8.060	8.207
17500	7.789	8.472	8.235	8.235	7.669
20000	7.325	8.604	8.070	8.849	8.721
22500	7.120	8.190	7.678	8.118	8.375
25000	6.317	8.219	8.905	9.066	8.024
27500	6.616	7.826	8.080	8.944	9.081
30000	7.011	8.030	8.208	9.066	6.964
35000	7.592	9.347	9.207	9.317	10.4
40000	7.566	8.762	8.050	9.367	9.227
45000	6.985	8.498	8.173	8.366	8.625

Table B.7: Goto results using 8 processors in GFlops

Appendix C

Developing an SRM interface for SRB

In this Appendix we present a more detailed and technically oriented discussion of our development of an Storage Resource Manager (SRM) interface for the SDSC Storage Resource Broker (SRB). This is aimed at documenting the project, and the issues we faced, to assist those who wish to create further SRM interfaces.

C.1 Use Cases

We present three use cases and their associated functional diagrams for the SRM2SRB system. These cover data retrieval from the SRB Master system, data retrieval from an SRB Server system and data insertion at the SRB Master.

C.1.1 Data Retrieval from SRB Master

Pre-conditions: Valid Grid Certificate, RLS file location
Post-conditions: Data is copied to user.

Main Success Scenario:

The user requests a file from the SRM2SRB interface. The system uses GMCat to retrieve the SRB format file location from the RLS file location. The system uses MCAT to authenticate and authorise the user. The system copies the file to the user.

Appendix C

Developing an SRM interface for SRB

In this Appendix we present a more detailed and technically oriented discussion of our development of an Storage Resource Manager (SRM) interface for the SDSC Storage Resource Broker (SRB). This is aimed at documenting the project, and the issues we faced, to assist those who wish to create further SRM interfaces.

C.1 Use Cases

We present three use cases and their associated functional diagrams for the SRM2SRB system. These cover data retrieval from the SRB Master system, data retrieval from an SRB Server system and data insertion at the SRB Master.

C.1.1 Data Retrieval from SRB Master

Pre-conditions: Valid Grid Certificate, RLS file location
Post-conditions: Data is copied to user.

Main Success Scenario:

The user requests a file from the SRM2SRB interface. The system uses GMCat to retrieve the SRB format file location from the RLS file location. The system uses MCAT to authenticate and authorise the user. The system copies the file to the user.

Extensions:

Authentication Failure - The user does not have a valid Grid Certificate. The system refuses request.

Authorisation Failure - The user has a valid Grid Certificate but it not authorised to access file. The system refuses request.

File Inaccessible - The requested file does not exist. The system refuses request.

GMCat/MCAT Inaccessible - File location, authentication, and authorisation unavailable. The system refuses request.

C.1.2 Data Retrieval from SRB node

Pre-conditions: Valid Grid Certificate.
Post-conditions: Data is copied to user

Main Success Scenario:

The user requests a file from the SRM2SRB interface which is not located locally. The system uses GMCat to retrieve the SRB format file location from the RLS file location. The system uses MCAT to authenticate and authorise the user. The system copies the file to the SRM2SRB node. The system copies the temporary file to the user.

Extensions:

Authentication Failure - The user does not have a valid Grid Certificate. The system refuses request.

Authorisation Failure - The user has a valid Grid Certificate but it not authorised to access file. The system refuses request.

File Inaccessible - The requested file does not exist. The system refuses request.

Node Inaccessible - The node containing the file is unavailable. The system refuses request.

Communications Failure - The transfer of data from the SRB Node to the SRB Master or from the SRB Master to the user is interrupted. The system attempts to re-transmit the data. The system returns error message.

C.1.3 Adding Data to SRB

Pre-conditions: Valid Grid Certificate.
Post-conditions: Data is stored. MCAT and RLS updated.

Main Success Scenario:

The user copies a file to the SRM2SRB interface. The system saves the file to SRB. SRB updates the MCAT and replicates file as necessary. GMCat updates the RLS as necessary.

Extensions:

Authentication Failure - The user does not have a valid Grid Certificate. The system refuses request.

Authorisation Failure - The user has a valid Grid Certificate but it not authorised to add file. The system refuses request.

C.2 SRM Functionality

This section outlines in broad terms each of the functions required by the SRM Version 2.1 (Final) Specification, their applicability to the SRM2SRB interface, and how they can be achieved using the SRB Jargon API.

C.2.1 Space Management Functions

srmReserveSpace

This function, which is intended to reserve space on the server prior to data transfer, has not been implemented by any SRM version so far. This is because the operating system is in charge of file management not the application layer. It is therefore possible for other applications on the machine to write to disk space which has been reserved or modify files already stored. It is unnecessary for an initial SRM2SRB system as the user is not allowed to write data to the SRB. A null value should be returned.

srmReleaseSpace

This function, which would release space reserved by *srmReserveSpace*, is unnecessary if space reservation is not implemented. There is no clear and obvious method of applying this to a distributed storage system where there are many variables to available storage space. A null value should be returned.

srmUpdateSpace

srmUpdateSpace is not needed, as its function, to update the metadata about available and reserved spaced, is not being supported. A null value should be returned.

srmCompactSpace

srmCompactSpace removes files that have been released by the client which required them. This reduces space allocated to the client to include only durable and permanent files and volatile files which are in use. This is intended to be used on systems which charge users for the storage space they use, rather than have allocated. As such it has more practical benefit to industry than particle physics.

srmGetSpaceMetaData

This function is used to retrieve information about the available space on a resource and the type of space available. This is only a concern when users are able to write to the system so SRM2SRB should return a null value to this query.

srmChangeFileStorageType

This function is used to move files between the three storage types of SRM, volatile, durable and permanent. While we may copy data from SRB to the local system if it is initially stored remotely, we do not need to provide the user the option of moving the data between the storage types as SRM2SRB is intended to be a retrieval system only. A null value should be returned.

srmGetSpaceToken

We do not need this function as space reservation is not being supported. A null value should be returned.

C.2.2 Permission Functions

srmSetPermission

srmSetPermission is used to change the permissions of a directory or file in the same manner as *chmod* does in Unix. It is an optional function in the SRM 2.1.1 specification. As users are not allowed to do this in SRM2SRB the function should return null. However, it would be possible using the SRB API to support this, given the user has authenticated using GSI. There is unlikely to be a need for this in particle physics applications but may potentially be for other disciplines.

srmReassignToUser

srmReassignToUser is used by SRM clients to grant read permission to another user. This is allowed by SRB, by the use of tokens, however is unsupported by SRM2SRB. This function is not necessary as users, if they have permission to read a file, are allowed to do this and no more.

srmCheckPermission

The SRM 2.1.1 specification calls for *srmCheckPermission* to check the local cache for the permissions associated with the specified file in relation to the user, and then check the permissions of the file at a remote site. For this function the role of SRM2SRB is to respond to other sites querying the permissions associated with a file.

This can be achieved by authenticating the user using GSI, translating the SRM URI to MCAT format using GMCat, and then querying the MCAT. The response should be in the *TReturnStatus* format.

C.2.3 Directory Functions

srmMkdir

This function is not needed as users are not able to write files into the initial version of SRM2SRB. In later versions this should map to the creation of a new sub-collection within the current one.

srmRmdir

The intended usage of SRM2SRB (Section 6.4.3) is to allow users to retrieve data from SRB while they are being migrated to SRM. Therefore, this function is not needed as users are not allowed to delete files or directories within the SRB system.

srmRm

This function is not needed as users are not allowed to delete files or directories within the SRB system.

srmLs

This function is not needed in the initial version of SRM2SRB because, the file metadata retrieved from RLS provides only a single URI. The system is therefore intended to only return a single file with each request. Future versions of the system can include this functionality by using the appropriate calls in the SRB Jargon API.

srmMv

The srmMv function has two use cases which we are concerned with: moving a file locally between directories; and moving a file from one machine to another. In the first case, SRM2SRB does not allow users to modify the location of data within SRB meaning this does not require implementing. In the second case, this would require deleting the file from SRB once it has been successfully copied to the second machine. As users are not allowed to delete files or directories, this function can be mapped to srmCopy.

C.2.4 Data Transfer Functions

srmPrepareToGet

srmPrepareToGet is the pull mode of SRM. It is intended to retrieve files from remote locations for the local user. We assume that there will not be any local users on the SRM2SRB system, so development should focus instead on *srmPrepareToPut*. This may be an issue for the system as remote users will be attempting to directly retrieve data using the GridFTP system (Section 6.4.3).

srmPrepareToPut

This function provides the push mode of SRM. We assume that once a request is made, SRM2SRB will ensure that the file is available locally e.g. retrieve from remote location if necessary using SRB. This can then be transfered to the remote location using GridFTP. This method can support further protocols so it may be possible to use the data transfer capabilities SRB already has for this task.

srmCopy

The *srmCopy* function can be used as both *srmPrepareToGet* and *srmPrepareToPut*. The specification calls for it to be applied to both files and directories. In the context of SRM2SRB we would prefer to limit this to individual files.

The function will need to: authenticate the user using MCAT, retrieve the file location in SRB format from GMCat, retrieve the file from SRB, request GridFTP transfer the file to the required location. The completion of this should be recorded in the metadata.

srmRemoveFiles

As users are not allowed to delete files this function is not required.

srmReleaseFiles

Volatile and durable space are not used in the SRM2SRB system so *srmReleaseFiles* is not needed. We assume that for early versions of the system file transfers which fail for what ever reason will be restarted from the beginning.

srmPutDone

This function is called by the user at end of transfer to send a confirmation of completion. SRM2SRB should use this to update the metadata database that the request has completed, at which point it should also delete any cached copy of the data.

srmAbortRequest

This function should cancel the request with the transfer software (typically GridFTP), delete the locally cached copy, if necessary, and update the MCAT and SRM2SRB metadata database on the final status of the request.

srmAbortFiles

This function is intended to selectively remove files from a multi-file request. As SRM2SRB is intended for individual file retrieval it should map to the *srbAbortRequest* function.

srmSuspendRequest

The initial version of SRM2SRB will not support the suspension of requests i.e. they will only return succeed or fail. This function does not therefore need to be supported. Future versions will be dependent on the version of GridFTP or other data transfer package being used.

srmResumeRequest

The initial version of SRM2SRB will not support the suspension of requests i.e. they will only return succeed or fail. This function does not therefore need to be supported. Future versions will be dependent on the version of GridFTP or other data transfer package being used.

srmStatusOfGetRequest

See *srmStatusOfCopyRequest.*

srmStatusOfPutRequest

See *srmStatusOfCopyRequest.*

srmStatusOfCopyRequest

srmStatusOfCopyRequest is called with the request ID, user ID and initial and final SURLs. This expands *srmStatusOfGetRequest* and *srmStatusOfPutRequest* which only require the inital and final SURL respectively. The responses are outlined in the SRM specification.

srmGetRequestSummary

This returns a summary of the status of the users requests. The responses are outlined in the SRM specification. It can be extended to return a summary of all the user's requests if the system is extended to support multiple sub-requests.

srmExtendFileLifeTime

Volatile and durable space are not used in the SRM2SRB system so *srmExtendFileLifeTime* is not needed. A null value should be returned.

srmGetRequestID

This function is used to retrieve the ID of either a single query by a user, or all queries by that user. While SRM2SRB is designed to retrieve individual files, it is possible for a user to submit simultaneous requests. As such the system will require a database of current queries which is outlined in Section C.3.

C.3 A Database for SRM2SRB

It is obvious from defining the requirements for the SRM functions, that the system requires a method of maintaining metadata about requests and temporary data locations. We can see two methods of achieving this, metadata files or a database. In either case we believe that the XML format is the most logical choice. There are many development tools available for creating and searching XML data efficiently.

The decision between the two potential solutions would have to be based on the assumed usage level, both of the number of users and the number of requests. We would generally recommend a database such as BerkeleyDB which supports XML in a native format, but this requires further clarification.

Appendix D

A hybrid client for Grid storage

One of our conclusions in Chapter 6 was that we need interoperability between the various Grid storage systems currently available. The best short term solution would be a new hybrid client. In this appendix we present the design and development of this. We then present a summary of the results we achieved testing the data transfer performance of the existing SRB client tools (Scommands) and an existing GridFTP (uberFTP) client. This is used to provide a comparison to the performance of our new client.

D.1 A new hybrid client

The Storage Resource Broker (SRB) and various implementations of the Storage Resource Manager specification, with many of these providing GridFTP as a transfer mechanism, are widely deployed in high energy physics research today. As development and research into making these systems inter-operable continues, there is a perceived need to provide a client which is capable of talking to both. In this section we discuss how we took an existing client, uberFTP, and added basic support for SRB. By basic support we mean fundamental operations such as authentication, connection management, file transfer and directory manipulation which are necessary and can be expanded on to provide more complex operations.

To do this we first discuss the current implementation of uberFTP and how we can add SRB support to it. We then present the design of our software and the dependencies it has in relation to the uberFTP code and SRB libraries. We will then discuss the implementation issues encountered.

D.1.1 uberFTP Design

UberFTP is a modular package comprising a GridFTP session handler, interactive interface and bindings to local commands. UberFTP is released as a GPT package. Within this package the source code files of interest to us (excluding Globus packaging information and compilation and deployment related files) are:

- *main.c* - This provides the global variables, such as the GridFTP connection, usage information about the options which can be called from the command line, the interpretation of these options with the getopt method, and an executor for batch commands. If a batch request has not been made by the user, the program calls its command interpreter which is in *cmds.c* and defined in *cmds.h*.

- *uberftp.h* - This library provides the definitions for booleans and return types for uberFTP e.g. success, failure, connection failure, etc.

- *cmds.c & cmds.h* - These files provide the prototypes of the commands available to the uberFTP client along with help and usage information. The source code file provides the instructions for the command interpreter and local operations. The commands for remote operations provide the necessary calls to the ftpSession functions.

- *ftpSession.c & ftpSession.h* - These files provide the FTP connection specific methods and prototypes.

D.1.2 Necessary SRB support

In adding support for SRB into uberFTP it is obvious that a minimal set of functionality is needed. This must be developed first and can then be built upon with more advanced functions in the future. This minimal set includes the ability to connect and disconnect from an arbitrary SRB server, authentication of the user, and the ability to upload and retrieve files. The ability to make and destroy directories (SRB collections) is also seen as being desirable.

Connecting to the SRB server requires a significant amount of information including the username, domain, home and server distinguished name for GSI connections. To make this easier for the user we believe the *MdasEnv* file should be used unless the user wishes to override it. This file is stored in the *.srb* subdirectory of the users

home directory and contains all the connection parameters needed. This is SRB's default method of storing and obtaining these values for the user.

We will therefore, start by adding the following functions to uberFTP:

- srbConnect - this command will connect to an SRB server. It should be able to do this using either user provided values or based on the values in the users' *MdasEnv* file.
- srbClose - this will terminate the connection to the SRB server.
- srbPwd - will provide the user with the directory they are currently working in.
- srbCd - will allow the user to return to their SRB home directory or to change the working directory.
- srbMkdir - will create a new directory / SRB collection.
- srbRmdir - will remove an empty SRB collection.
- srbPut - will copy a file from the local system to the current working directory on the SRB server.
- srbGet - will retrieve a file from the present SRB working directory and store it on the local machine.

There is the potential for several dependencies to exist between these methods which in development we will try to minimise. All, with the obvious exception, will need an existing connection to the SRB server to work. It would also be useful for us to maintain a shared value for the present working directory (PWD) in our library. Each of the methods which use this (srbPwd, srbCd, srbMkdir, srbRmdir, srbPut and srbGet - and srbConnect which will set it initially) could take this PWD as a parameter however it is as easy for us to maintain this as a global value.

UberFTP has an interactive client, which is called *CmdInterpreter* in the *cmds* files, that allows the user to execute commands in an arbitrary manner. This is done with a command line interface in much the same way as a typical FTP client. We would like the commands we develop to also be available in this way. The current uberFTP version can also perform prepared, or batch, operations using the

ExecuteCmd method from the Unix command line. This allocates memory, ensures that a valid method is being requested and then executes it. We would also like to provide our methods in this way.

D.1.3 Developing SRB support within uberFTP

In this section we present the actual development of our code including the changes we made to existing source code files in uberFTP and the files we created to support SRB access.

We took the latest release of the uberFTP client (1.13) which is packaged as a gzipped tar ball and unpackaged it. We ran the configuration script (*configure*) with the options '–with-globus-flavor=gcc32dbg' and '–with-globus=PATH'. This generated a Makefile based on our installation. Running *make* and *make install* will compile the client and install it. We also used '–prefix=$HOME' so that the binaries would not be installed in the default location of '/usr/local/'.

The *Makefile* for the uberFTP client proved to be one of the most problematic areas of development. When running it several dozen error messages about undefined variable types in one of the SRB libraries were produced. After long discussion with developers of both SRB and uberFTP the error messages were tracked down to the lack of the operating system name being passed as a parameter to the compiler. By adding the option -DPORTNAME_linux to the DEFS parameter, the SRB libraries were able to configure the type values correctly allowing us to compile our software into the existing client. This has been fed back to the SRB and UberFTP developers.

Next we adapted *main.c* where we added several lines to the *Usage* method to include our '-s' option for SRB batch connections and instructions on its use. We added our '-s' option to the *getops* loop. This sets a flag (SRBFLAG). Assuming the standard SRB user environment file is available this will then be used to connect to the SRB server, run the command given to it and then closes the connection. In total this added 10 lines to the file and does not change any of the initial uberFTP functionality or usage.

We added the prototypes for our methods to *cmds.h* to maintain consistency, as this represent only 9 additional lines. We also need to adapt *cmds.c* by adding

information about our commands into the array, *CmdList*. This array contains a method name, the method name the user sees and a comment about its function and permitted parameters. This and the include statement for our *srbSession.c* file added approximately 10 lines.

The current version of the srbSession.c file is approximately 500 lines of code covering 9 methods, additions to existing files and global parameters. The initial version of uberFTP compiled on our machine was only 92KB in size, although it does make extensive use of dynamically linked libraries. The version we created with SRB support was 2.6MB, mainly due to the inclusion of the SRB libraries. While both storage and system memory are cheap, future work should consider attempting to optimise the SRB libraries used. This does not mean that the memory footprint of the executable has increased, as the libraries are only loaded when necessary. However we would suggest that users share the binary rather than creating a release in their home directory.

D.1.4 Initial testing of the hybrid client

We tested each of the methods we had added to uberFTP in both batch and interactive mode. These tests included error checking for invalid parameters, missing or failed connections, the SRB server being unavailable and so on. They showed that for an invalid number of commands parameters being passed to each of the methods the code is robust and can recover. When a connection to the SRB server is not available the functions are able to realise this and instruct the user that one is needed. Connection failures during operation can be detected by each of the methods. These produce an error message for the user using *stderr* and an appropriate return value for uberFTP. Obviously invalid values in the parameters, such as requesting more than 20 data streams or less than one data stream, is noticed and the user is requested to provide more reasonable parameters.

D.2 Performance Testing

We are interested in testing the transfer performance of the SRB client, uberFTP and our hybrid client when we vary the file size and the number of parallel data streams. We are interested in testing the different clients because different transfer mechanisms, such as the parallel I/O implementation used by SRB and the GridFTP

protocol used by uberFTP, have different characteristics as they are intended for different situations. By testing the SRB client software we will also see if our methods in the hybrid client are reducing transfer performance.

We are interested in two scenarios for all the clients, their performance on a single machine and their performance when the client and the server are run on separate machines. In the first scenario we are limited only by the physical hardware so we will see the raw performance of the various systems. In the second scenario we know we have a physical limitation, the network, and wish to see how the systems cope with this.

The reason we are interested in the effect of varying the file size is because with very small files the system does not have the opportunity to make use of its resources in an optimal way. This means that while from the users perspective the real time, or wall clock time, taken for the data transfer is very small, the actual transfer rate is very poor. With large files a different problem presents itself. In this case the memory buffer used by the system will eventually becomes saturated, and more processing time must be allocated to swapping data from the storage system to memory than is used to manage its transfer. This typically reduces the transfer rate.

Theoretically, increasing the number of parallel data streams should increase the transfer rate in a linear fashion. However, we know that for each additional data stream there is a greater than linear increase in processing overhead. There is both the extra data stream to manage and synchronisation between the streams to consider. In addition to this, there is a limit to the storage systems transfer performance and capacity of the system memory. We know from our work in Section 5.2.4 that even relatively high performance processors are unable to make full use of Gigabit Ethernet, so in performing these tests we will show whether client machines should concern themselves with parallelism.

As we said in Section 6.3.1, SRB is a monolithic system that may also use remote storage resources in addition to the primary system. We will test this scenario to see what overhead having to communicate with two systems in parallel has on transfer performance.

D.2.1 UberFTP

Local Machine

Tests were performed on a machine[1] which was running GridFTP as *root* and uberFTP from a client account. We tested file upload and download with file sizes of 1KB, 1MB, 10MB, 100MB and 1GB varying the number of data streams used. Each test was run a minimum of 100 times and the mean results of the transfer rate and their standard deviation is presented in Tables D.1 and D.2. The single stream results, which are representative of those recorded, are presented in Figure D.1. The figure presents the read (get) performance in red and the write (put) performance in blue.

We also looked into varying the size of the user proxy, however, this had no noticeable impact on performance so in the interests of space we have not included these results.

As we can see in Table D.1, the performance at each file size is consistent, within the standard deviation, when varying the number of streams. This suggests that either the uberFTP client is overriding the instruction on what parallelism to use, or the physical performance limits have been reached, or the overhead from having multiple data stream cancels out any performance gain. After consultation with the uberFTP developers and a code review, we discounted the idea that the system was failing to accept the instruction to use a specific number of parallel streams. If the maximum physical performance was being reached we would expect the data rates at 10MB, 100MB and 1GB to be roughly consistent and yet they are not. This leaves us with the conclusion that on the local system increasing the number of data stream to be written to hard disk has little or no effect.

In Table D.2 we see that at the 10MB and 100MB file sizes the download performance is significantly better than the upload performance. This can be attributed to the hardware with the storage system, i.e. the hard disk, being optimised for data retrieval. We believe that the drop in performance at a file size of 1GB is a lack of adequate memory for buffering by the client and the storage system. This contrasts with the upload performance in Table D.1 where the client needs only to stream the data from disk to the server.

[1] wn2.epcc.ed.ac.uk

With the 100MB file size we can see a very large standard deviation for downloads, significantly higher than recorded at any other file size. This is not due to a few results skewing the sample. We reran the test with a sample size of 500 and re-calculated the result. This error continues even with an expanded sample. We believe it to be due to the size of the buffers used by uberFTP. The best explanation is that the buffers are adequate for smaller file sizes. For larger files the processor is swamped by the need to swap data in and out of memory which lowers the data rate and reduces the deviation.

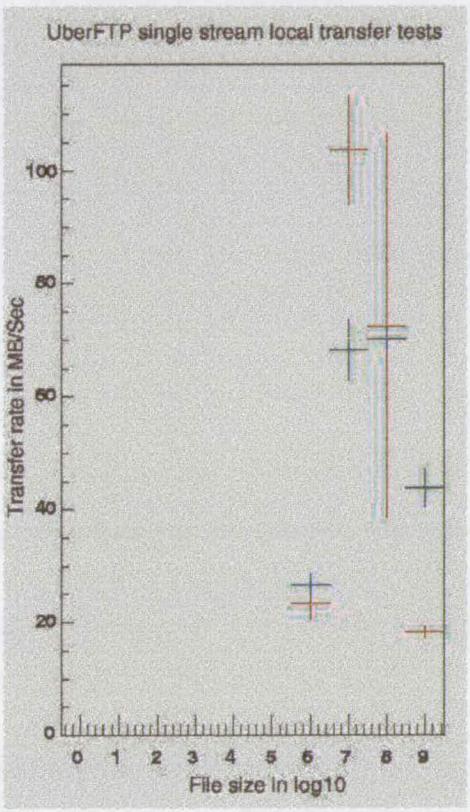

Figure D.1: UberFTP client local read / write performance

Streams	1	2	5	10	20
1KB file	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)
1MB file	26.68 (2.05)	26.91 (2.51)	26.54 (2.29)	26.62 (2.24)	26.14 (3.03)
10MB file	68.2 (5.36)	68.32 (6.04)	65.36 (8.59)	60.46 (4.03)	58.76 (6.36)
100MB file	70.16 (2.19)	70.54 (3.89)	71.14 (6.41)	70.43 (2.36)	69.27 (5.37)
1GB file	43.86 (3.33)	43.76 (1.79)	43.66 (1.96)	44.03 (1.79)	43.92 (1.71)

Table D.1: UberFTP File Upload (in MB/sec) using a 512 bit proxy on local machine

Streams	1	2	5	10	20
1KB file	0.02 (0.00)	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)
1MB file	23.46 (2.92)	26.78 (0.26)	26.44 (2.83)	26.73 (0.77)	26.66 (0.68)
10MB file	103.6 (9.58)	98.24 (4.21)	99.51 (4.04)	97.97 (3.33)	96.26 (5.7)
100MB file	85.4 (4.1)	69.78 (3.87)	66.69 (4.28)	73.44 (3.51)	71.21 (5.41)
1GB file	18.23 (0.9)	20.05 (1.21)	20.61 (1.27)	20.56 (1.2)	20.62 (1.31)

Table D.2: UberFTP File download (in MB/sec) using a 512 bit proxy on local machine

Remote Machine

We re-ran the tests using the uberFTP client on a remote machine[2] on the same LAN. We tested file upload and download with file sizes of 1KB, 1MB, 10MB, 100MB and 1GB using a 512 bit grid proxy and varying the number of data streams. Each test was run 100 times and the mean results of the transfer rates and their standard deviations are presented in Tables D.3 and D.4. Figure D.2presents the read (get) performance in red and the write (put) performance in blue of the single data stream results.

As we can see from Figure D.2 the performance over the network for both read and write operations is lower than when these tests were conducted on the local machine. Again we see the best performance at the 10MB file size for both reading and writing. As we saw on the local machine tests in Figure D.1, write performance is almost identical for the 10 and 100MB file size. The 100MB download performance has a high standard deviation again. As with the tests on the local machine we reran this with a sample size of 500. From this we can state that this is not due to a few random results at the upper and lower ends of the results but due to a genuinely large distribution.

Streams	1	2	5	10	20
1KB file	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	0.02 (0.00)	0.01 (0.00)
1MB file	24.49 (2.03)	23.78 (2.60)	21.66 (3.55)	15.62 (1.79)	9.81 (1.09)
10MB file	54.23 (6.95)	47.92 (3.37)	50.23 (4.83)	47.03 (2.83)	41.01 (4.03)
100MB file	48.5 (10.55)	27.69 (0.9)	30.72 (1.29)	31.31 (1.48)	30.83 (1.32)
1GB file	45.6 (3.69)	27.68 (0.28)	31.06 (0.33)	32.92 (0.4)	34.58 (0.64)

Table D.3: UberFTP File Upload (in MB/sec) using a 512 bit proxy to a remote machine

[2] glenlivet.epcc.ed.ac.uk

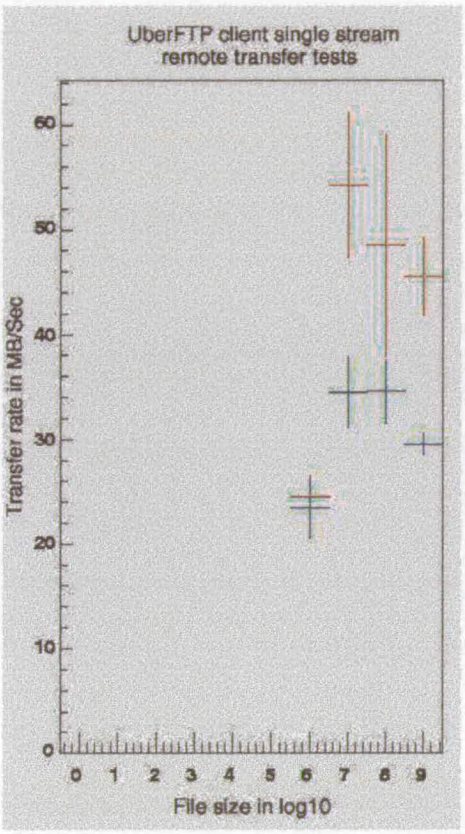

Figure D.2: UberFTP client remote read / write performance

Streams	1	2	5	10	20
1KB file	0.02 (0.00)	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)	0.03 (0.00)
1MB file	23.46 (2.92)	26.06 (2.97)	26.17 (2.21)	25.45 (3.49)	24.97 (2.48)
10MB file	34.57 (3.42)	38.72 (2.17)	39.69 (2.24)	39.6 (2.99)	39.77 (2.33)
100MB file	34.59 (3.07)	37.25 (3.02)	38.18 (2.38)	37.85 (2.63)	38.08 (2.23)
1GB file	29.56 (1.01)	28.05 (1.21)	28.61 (1.27)	27.56 (1.2)	28.62 (1.31)

Table D.4: UberFTP File Download (in MB/sec) using a 512 bit proxy from a remote machine

D.2.2 SRB Scommands

The SRB Scommands are packaged with the SRB Server software and provide Unix-like commands (e.g. Sls, Smkdir, etc) for authenticating the user, copying files to, and retrieving files from, the server and various other tasks. In these tests we look specifically at the Sput and Sget commands which allow the user to upload and download files with a user defined number of parallel streams. If the number of streams is not specified by the user SRB will attempt to decide the number of streams to use based on file size. This is typically set to be one stream per 30 Megabytes of file size.

Local Machine

Tests were performed on a machine[3] which was running the SRB server as *srbAdmin* and the Scommands suite from a client account. We tested file upload and download with file sizes of 1KB, 1MB, 10MB, 100MB and 1GB using varying numbers of data streams. Each test was run 100 times and the mean results of the transfer rates achieved are presented in Tables D.5 to D.6 and Figure D.3. Again we also looked at varying the key length of the grid proxy, but this had no significant effect on performance. The figure presents the read (get) performance in red and the write (put) performance in blue.

As we can see from Figure D.3 there is a clear performance increase as file size increases for both read and write operations which falls after 100MB. This is within our expectations for the local storage system performance. There is a far lower standard deviation for our 100MB file size tests here which indicates better buffer usage by SRB when compared to uberFTP.

Streams	1	2	5	10	20
1KB file	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)
1MB file	4.35 (0.16)	4.32 (0.21)	4.27 (0.27)	4.26 (0.29)	4.26 (0.19)
10MB file	18.69 (1.27)	19 (0.73)	18.65 (1.62)	18.54 (1.23)	18.63 (0.99)
100MB file	27.69 (2.04)	27.83 (1.60)	27.84 (1.70)	27.74 (1.67)	27.80 (1.47)
1GB file	19.9 (1.7)	18.78 (1.01)	20.65 (1.62)	19.54 (1.3)	19.63 (1.09)

Table D.5: SRB File Upload (in MB/sec) using a 512 bit proxy from local machine

[3] wn2.epcc.ed.ac.uk

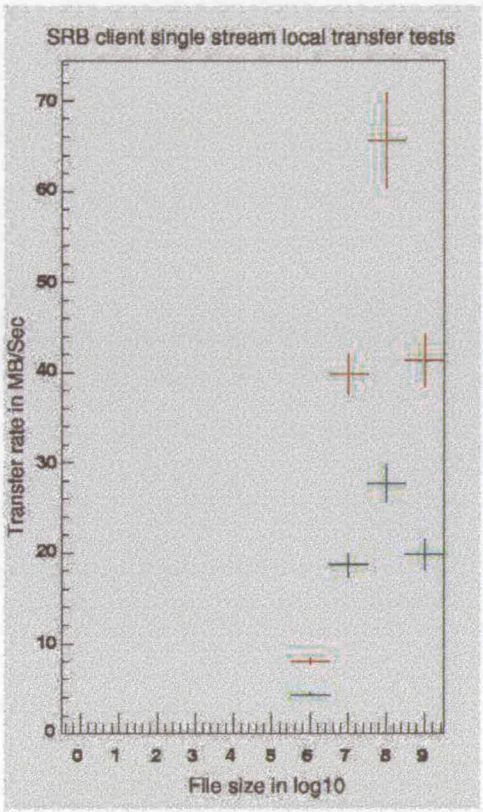

Figure D.3: SRB client local read / write performance

Streams	1	2	5	10	20
1KB	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)
1MB	8.05 (0.42)	8.10 (0.05)	8.10 (0.05)	8.09 (0.05)	8.05 (0.3)
10MB	39.86 (2.27)	40.06 (0.34)	40.05 (0.32)	39.98 (0.39)	40.04 (0.31)
100MB	65.57 (5.31)	65.66 (3.71)	66.46 (1.12)	66.17 (1.30)	66.09 (1.93)
1GB	41.35 (2.93)	41.55 (1.33)	41.63 (1.19)	41.68 (1.06)	41.48 (1.19)

Table D.6: SRB File Download (in MB/sec) using a 512 bit proxy from local machine

Remote Machine

The tests were performed between two machines[4] with WN2 running the SRB server as *srbAdmin* and the Scommands suite being run from a client account on *glenlivet*. We tested file upload and download with file sizes of 1KB, 1MB, 10MB, 100MB and 1GB. Each test was run 100 times and the mean results and their standard deviations presented in Tables D.7 and D.8. Figure D.4 presents the read (get) performance in red and the write (put) performance in blue.

Again we can see a peak performance at 100MB and a bell curve for the download results. The results recorded are lower than seen in the local machine tests which suggests that the network bandwidth is a limiting factor. The 1GB file size write results are interesting as they are higher than for the 100MB file size and this behaviour has not been seen before. We expanded the test to 200 data points and recorded the same result. We also tested both the 100MB and 1GB performance from a different machine on the same LAN and saw the same result. We attribute this to adequate network bandwidth and memory buffers and point out that the performance is still lower than achieved when testing on a single machine.

Streams	1	2	5	10	20
1KB file	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)
1MB file	3.88 (0.52)	3.90 (0.49)	3.89 (0.49)	3.82 (0.59)	3.82 (0.5)
10MB file	14.61 (0.97)	14.8 (1.05)	14.62 (0.98)	14.62 (1.06)	14.46 (1.13)
100MB file	21.22 (0.9)	21.5 (1.15)	21.44 (1.1)	21.28 (0.86)	21.29 (1.00)
1GB file	23.11 (0.76)	23.11 (0.29)	23.12 (0.29)	23.18 (0.3)	23.13 (0.28)

Table D.7: SRB File Upload (in MB/sec) using a 512 bit proxy to a remote machine

Streams	1	2	5	10	20
1KB file	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)
1MB file	5.61 (0.39)	5.65 (0.52)	5.68 (0.12)	5.68 (0.13)	5.69 (0.12)
10MB file	23.61 (1.18)	23.63 (1.01)	23.77 (0.46)	23.65 (0.9)	23.65 (1.47)
100MB file	34.09 (3.22)	34.55 (2.72)	34.59 (2.03)	34.33 (3.65)	34.55 (3.92)
1GB file	13.07 (1.42)	12.76 (1.35)	12.74 (1.41)	12.16 (1.40)	12.81 (1.38)

Table D.8: SRB File Download (in MB/sec) using a 512 bit proxy from a remote machine

[4] wn2.epcc.ed.ac.uk and glenlivet.epcc.ed.ac.uk

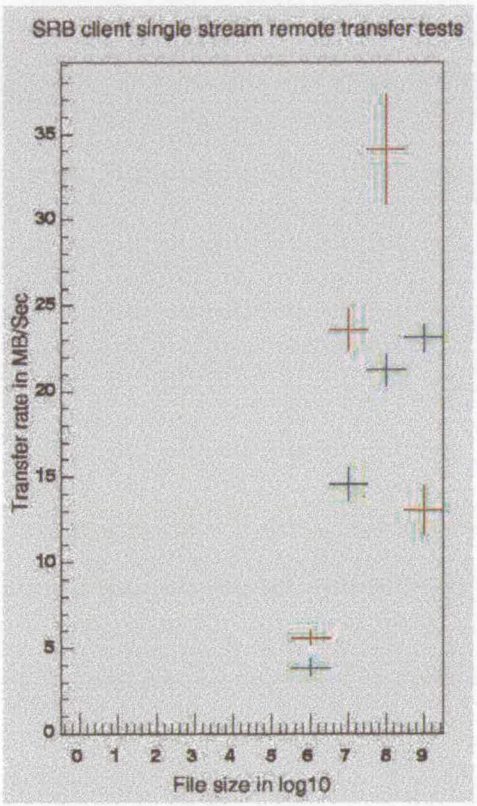

Figure D.4: SRB client remote read / write performance

3rd Party Machine

SRB differs from GridFTP in that it is a distributed system which can include multiple machines acting as storage nodes. In this test we upload and retrieve data from a machine other than the one which performs authentication and authorisation. While we are not able to compare the results of these tests with GridFTP we are able to see what the effect of this 3rd party transfer is on performance. In these tests the client and SRB Master (SRB server + MCAT) we hosted on wn2.epcc.ed.ac.uk with remote storage on glenlivet.epcc.ed.ac.uk.

The results of our tests are presented in Tables D.9 and D.10 and summarised in Figure D.5. The figure presents the read (get) performance in red and the write (put) performance in blue.

In these tests we can see similar performance to that seen in Figure D.3, with read and write performance increasing to the 100MB file size point and then decreasing. The decrease in performance at 1GB is less than we saw in the local system tests. This we attribute to the network buffering data.

Streams	1	2	5	10	20
1KB	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)
1MB	4.68 (0.34)	4.66 (0.19)	4.61 (0.31)	4.59 (0.34)	4.58 (0.26)
10MB	19.56 (0.96)	19.67 (1.11)	19.52 (1.05)	19.56 (0.71)	19.30 (1.22)
100MB	27.86 (2.00)	27.83 (1.88)	27.88 (2.01)	27.76 (1.94)	27.84 (1.84)
1GB	27.64 (0.82)	27.63 (0.86)	27.87 (0.80)	27.67 (0.91)	28.03 (0.72)

Table D.9: SRB File Upload (in MB/sec) using a 512 bit proxy to a 3rd party machine

Streams	1	2	5	10	20
1KB	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)	0.01 (0.00)
1MB	7.07 (0.4)	7.12 (0.02)	7.11 (0.03)	7.11 (0.05)	7.12 (0.03)
10MB	38.98 (2.22)	39.20 (0.23)	39.32 (0.22)	39.15 (0.3)	39.31 (0.24)
100MB	71.34 (5.72)	71.5 (2.08)	71.82 (1.81)	72.18 (1.53)	71.70 (3.22)
1GB	41.04 (3.34)	41.41 (1.36)	41.63 (1.08)	41.48 (1.3)	41.64 (1.04)

Table D.10: SRB File Download (in MB/sec) using a 512 bit proxy from 3rd party machine

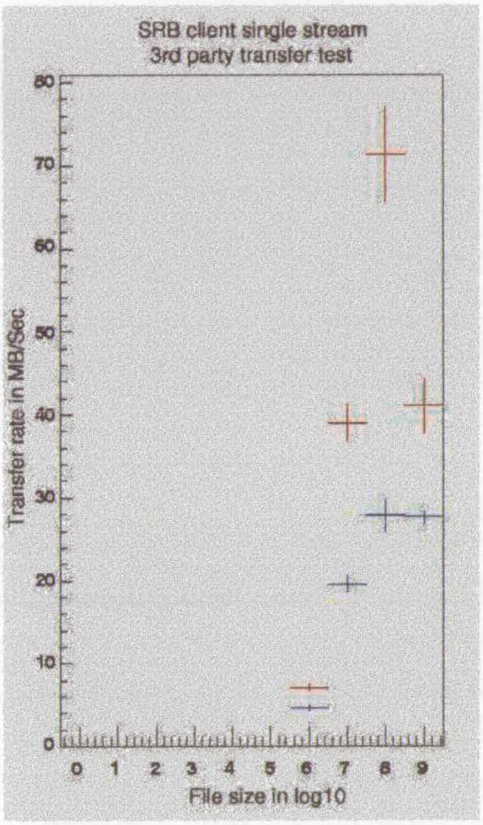

Figure D.5: SRB client 3rd party read / write performance

D.2.3 Hybrid Client

Local Machine

We tested the transfer performance of our hybrid client on a machine which was also running an SRB server. This was done by using the non-interactive, or batch, method where the request is passed to the uberFTP client at the command line and after execution the client exits to the Unix shell. Using a grid proxy certificate of 512 bits we tested put and get operations with a 1KB, 1MB, 10MB, 100MB and 1GB sized file varying the number of streams used for transfer. The results of these tests are presented in Tables D.12 and D.11 and Figure D.6. The figure presents the read (get) performance in red and the write (put) performance in blue.

As we can see the read performance (get) is higher than the write performance. This was expected based on our knowledge of the hardware in use. The variance in the performance of the read operations at 100MB is interesting as it is not repeated at the 1GB level. This could be due to a variety of hardware and operational issues.

Streams	1	2	5	10	20
1KB	0.03 (0)	0.03 (0)	0.03 (0)	0.03 (0)	0.03 (0)
1MB	2.75 (0.27)	2.76 (0.15)	2.66 (0.19)	2.65 (0.03)	2.57 (0.26)
10MB	14.07 (0.69)	11.52 (0.7)	12.69 (0.9)	12.27 (1.75)	12.57 (1.25)
100MB	24.61 (4.17)	27.05 (4.24)	26.65 (5.54)	27.08 (5.54)	26.59 (5.62)
1GB	12.76 (0.39)	11.42 (0.12)	9.67 (0.12)	26.22 (7.39)	35.47 (3.15)

Table D.11: Hybrid client file upload (in MB/sec) using a 512 bit proxy from local machine

Streams	1	2	5	10	20
1KB	0.03 (0)	0.03 (0)	0.03 (0)	0.03 (0)	0.03 (0)
1MB	3.42 (0.08)	2.61 (0.82)	1.58 (1.02)	3.34 (0.11)	3.33 (0.11)
10MB	28.11 (2.34)	27.41 (1.55)	25.2 (2.56)	26.8 (10.78)	23.65 (4.11)
100MB	45.05 (6.23)	48.8 (0.98)	41.12 (2.52)	38.47 (4.12)	28.88 (9.91)
1GB	14.42 (0.06)	16.84 (0.85)	12.98 (2.12)	10.45 (2.69)	10.77 (2.24)

Table D.12: Hybrid client file download (in MB/sec) using a 512 bit proxy from local machine

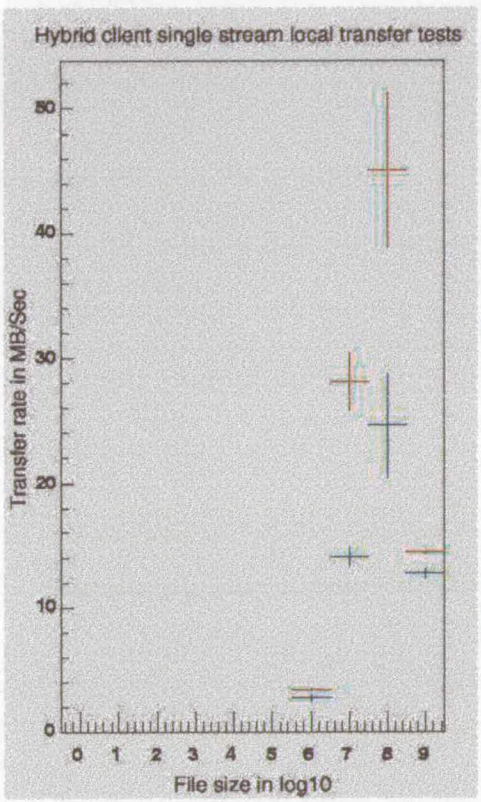

Figure D.6: Hybrid client local party read / write performance

Remote Machine

We tested the batch operation of our hybrid client on a machine which was separate to the one running the SRB server[5]. Using a grid proxy certificate of 512 bits we tested put and get operations with a 1KB, 1MB, 10MB, 100MB and 1GB sized file varying the number of streams used for transfer. The results of these tests are presented in Tables D.13 and D.14. Figure D.7 presents the read (get) performance in red and the write (put) performance in blue.

This performance is broadly similar to the remote SRB performance shown in Figure D.4 upto the 100MB file size. The drop in performance at the 1GB file size suggests that if we were to use our client for very large file transfers we should look at improving its use of memory buffers.

The large standard deviation at the 100MB file size for downloading data was tested further. Again it appears to be an issue with the size of the file rather than random outlying results.

Streams	1	2	5	10	20
1KB	0.003 (0)	0.003 (0)	0.003 (0)	0.003 (0)	0.003 (0)
1MB	2.48 (0.08)	2.55 (0.09)	2.55 (0.08)	2.45 (0.09)	2.39 (0.06)
10MB	11.79 (0.83)	11.54 (0.67)	11.83 (0.7)	11.77 (0.83)	11.52 (0.8)
100MB	23.01 (2.38)	23.01 (1.85)	22.89 (1.23)	22.89 (1.31)	22.71 (1.03)
1GB	11.81 (0.20)	12.33 (0.31)	11.77 (0.19)	11.27 (1.35)	11.51 (0.3)

Table D.13: Hybrid client file upload (in MB/sec) using a 512 bit proxy from remote machine

Streams	1	2	5	10	20
1KB	0.003 (0)	0.004 (0)	0.004 (0)	0.004 (0)	0.004 (0)
1MB	3.45 (0.24)	3.39 (0.45)	3.48 (0.30)	3.47 (0.2)	3.50 (0.13)
10MB	22.24 (2.87)	22.82 (1.93)	22.62 (1.54)	22.31 (2.39)	22.4 (2.28)
100MB	41.66 (11.46)	23.66 (1.61)	22.7 (2.89)	22.5 (2.46)	22.7 (2.58)
1GB	34.97 (1.96)	24.66 (0.66)	22.03 (0.52)	22.14 (0.36)	21.12 (0.37)

Table D.14: Hybrid client file download (in MB/sec) using a 512 bit proxy from remote machine

[5] glenmorangie.epcc.ed.ac.uk

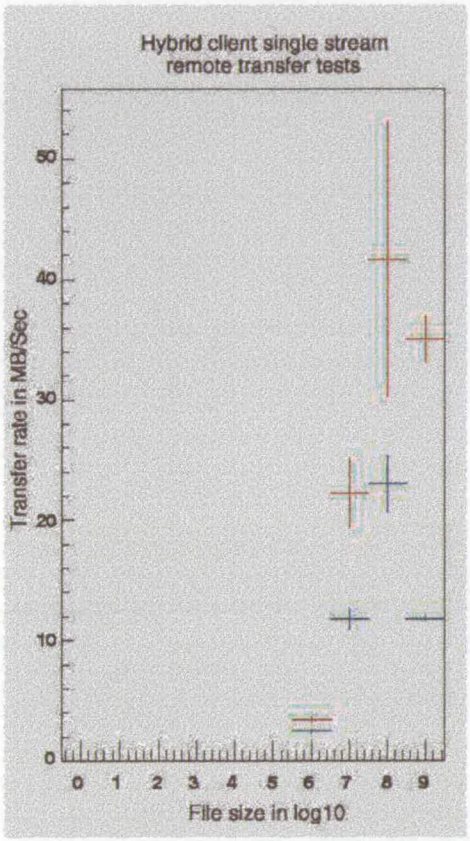

Figure D.7: Hybrid client remote party read / write performance

D.2.4 Performance Test Conclusions

The tests we conducted showed several interesting results. In all the tests there appears to be a "sweet spot". This is where performance is maximised as buffers are not yet saturated and the file size is large enough to minimise the effect of overhead. However this point is not the same for both systems. The results from the uberFTP tests suggest that its best performance can be achieved in the 10 - 100MB file size range. SRB and our hybrid client in contrast appear optimal at the 100MB file size.

As we would expect, in almost all the tests conducted, with the exception of some uberFTP tests (which we will discuss shortly) the file retrieval performance was better than the file upload, or writing, performance. This is due to the performance characteristics of the storage system. The performance of the network has at least as important a role as the local system. This is especially noticeable in the uberFTP tests (Figures D.1 and D.2) where the effective performance is halved once the network element is introduced.

The results of the 3rd party tests with SRB are interesting as they show slightly better performance than the direct, remote connection tests. This was unexpected because in these tests the system running the SRB client is having to communicate with both the SRB Master, where information is recorded in the MCAT database, and the remote SRB storage system. We believe this performance difference is due to the mixed configuration of systems on our LAN which we discussed in Chapter 5. Specifically, we have systems configured for both half-duplex and full-duplex operation and this is the likely cause of the performance difference. This was discussed in Section 5.2.4. This is a factor which is outside our control.

From the results of the tests we conducted on our hybrid client we can see that its "sweet spot" is the same as SRB's at approximately 100MBs. We actually recorded better performance for the 1GB tests compared to the SRB tests but believe this to be due to the network configuration. Specifically, we used a different remote machine for these tests, whose network configuration is the probable source of this difference. This change in machine was due to resource reallocation and maintenance by ScotGrid as part of LCG. Although both machines are on the same subnet one is configured for half-duplex and the other for full-duplex operation which we discussed in Chapter 5.

The performance of our client is acceptable for the type of scenario we can imagine it being used in i.e. occasional medium sized file movement. It is not as optimised as the other two packages but this is due more to its lack of maturity than any inherent problem.

D.3 Conclusion

To summarise, we created a GridFTP testbed by installing the latest release of the GridFTP server and uberFTP client. We then created a multiple node SRB system. We extended the uberFTP GridFTP client package to support basic SRB client functionality. In doing this we discovered and resolved several issues with the SRB installation routines and uberFTP interoperability with the latest release of the ANL GridFTP server. All three systems were tested to discover data file transfer performance in a variety of situations and compared. The performance and reliability of our hybrid client was adequate for our intended usage scenario and provides a sound basis for future development work.

Bibliography

[1] The ALICE experiment homepage.
http://alice.web.cern.ch/Alice/.

[2] The AstroGrid project homepage.
http://www.astrogrid.org.

[3] The ATLAS experiment homepage.
http://atlas.web.cern.ch/Atlas/.

[4] Automatically Tuned Linear Algebra Software (ATLAS).
http://math-atlas.sourceforge.net/.

[5] BaBar Analysis Tools.
http://www.slac.stanford.edu/BFROOT/www/Computing-
/Offline/AnalysisTools/AnalysisTools.html.

[6] BaBar Event Store Data Management.
http://www.slac.stanford.edu/BFROOT/www/Public/Computing-
/Databases/users/dataManagement.shtml.

[7] BaBar MonteCarlo Production homepage.
http://www.slac.stanford.edu/BFROOT/www/Computing/Offline/Production/.

[8] BABAR Online Event Processing.
http://www.slac.stanford.edu/BFROOT/www/Computing/Online-
/EventProc/newOep.html.

[9] The BBGUtils Package.
http://www.slac.stanford.edu/~aearl.

[10] BdbServer, A Data Distribution Server.
http://www.slac.stanford.edu/BFROOT/www/Computing/Offline-
/DataDist/BdbServer.html.

[11] BLAS Frequently Asked Questions.
http://www.netlib.org/blas/faq.html.

[12] Cable and Wireless Marine: A History.
http://www.cwhistory.com/history/html/CWMarine.html.

[13] Cable and Wireless Worldwide homepage.
http://www.cw.com.

[14] The Cancer Busters project.
http://www.grid.org/projects/cancer/.

[15] The CCLRC SRB Guide.
http://www.e-science.clrc.ac.uk/web/projects/storage_resource_broker.

[16] The Climate Prediction project.
http://www.climateprediction.net/index.php.

[17] CM2 - An Introduction.
http://www.slac.stanford.edu/BFROOT/www/Computing-
/Documentation/CM2/intro/.

[18] The CMS experiment homepage.
http://cmsinfo.cern.ch/Welcome.html/.

[19] The CORBA homepage.
http://www.corba.org.

[20] The Cray homepage.
http://www.cray.com/.

[21] The DataGrid Project - EU Deliverables.
http://eu-datagrid.web.cern.ch/eu-datagrid/EUDocuments-
/Deliverables/default.htm#Overall project reports.

[22] The Dublin Core Metadata Initiative homepage.
http://dublincore.org/.

[23] Europe Exceeds U.S. in Refining Grid Computing, NYTimes.
http://query.nytimes.com/gst/abstract.html?
res=F60614F83A5D0C738DDDA80994DB404482.

[24] The European DataGrid homepage.
http://www.eu-datagrid.org.

[25] European Grid for Enabling EScience homepage.
http://egee-intranet.web.cern.ch/egee-intranet/gateway.html.

[26] The Folding at Home project.
http://www.stanford.edu/group/pandegroup/folding/.

[27] Frequently Asked Qustions on the Web Services Resource Framework.
http://www.globus.org/wsrf/faq.asp.

[28] Geant 4 homepage.
http://wwwasd.web.cern.ch/wwwasd/geant4/geant4.html.

[29] The Geant website.
http://www.dante.net/server/show/nav.007&.

[30] GGF GridFTP Working Group.
http://www.gridforum.org/6_DATA/gridftp.htm.

[31] The Global Grid Forum homepage.
http://www.gridforum.org.

[32] Globus Security Overview.
http://www.globus.org/security/overview.html.

[33] Globus Toolkit 3 factsheet.
http://www.globus.org/toolkit/gt3-factsheet.html.

[34] Globus Toolkit Public Licence.
http://www-unix.globus.org/toolkit/license.html.

[35] The Grid Physics Network homepage.
http://www.griphyn.org.

[36] The GridFTP Protocol and Software.
http://www-fp.globus.org/datagrid/gridftp.html.

[37] GridFTP: Universal Data Transfer for the Grid.
http://www-fp.globus.org/datagrid/deliverables/C2WPdraft3.pdf.

[38] High-Performance BLAS by Kazushige Goto.
http://www.cs.utexas.edu/users/kgoto/.

[39] Home Page of the RooFit Toolkit for Data Modeling.
http://www.slac.stanford.edu/BFROOT/www/Computing-
/Offline/ROOT/RooFit/index.html.

[40] The HPCx homepage.
http://www.hpcx.ac.uk/.

[41] HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers.
http://www.netlib.org/benchmark/hpl/.

[42] IBM zSeries.
http://www-1.ibm.com/servers/eserver/zseries/.

[43] The IETF PKI Working Group homepage.
http://www.ietf.org/html.charters/pkix-charter.html.

[44] The International DataGrid Laboratory homepage.
http://www.ivdgl.org.

[45] The KANGA homepage.
http://www.slac.stanford.edu/BFROOT/www/Computing-
/Offline/Kanga/index.html.

[46] Kanga: the ROOT of all ...
http://www.slac.stanford.edu/BFROOT/www/doc/workbook-
/kanga1/kanga1.html.

[47] LCG User Registration.
http://lcg-registrar.cern.ch/virtual_organization.html.

[48] LHC Computing Grid homepage.
http://lcg.web.cern.ch/LCG/.

[49] The LHCb experiment homepage.
http://lhcb.web.cern.ch/lhcb/.

[50] Linux DCE, CORBA and DCOM Guide.
http://linas.org/linux/corba.html.

[51] The MetaData Catalog - MCAT.
http://www.npaci.edu/SRB/guide/FAQs.html#mcat.

[52] The MetaData Catalog Guide.
http://www.npaci.edu/DICE/Software/SRB/mcat.html.

[53] Microsoft DCOM Technical Overview.
http://msdn.microsoft.com/library/default.asp?-url=/library/en-us/dndcom/html/msdn_dcomtec.asp.

[54] Microsoft Distributed Component Model.
http://www.microsoft.com/com/tech/DCOM.asp.

[55] The Minos experiment homepage.
http://www-numi.fnal.gov/.

[56] The National e-Science Centre.
http://www.nesc.ac.uk.

[57] National Grid Service - The UK's core production computational and data grid.
http://www.ngs.ac.uk/.

[58] NorduGrid homepage.
http://www.nordugrid.org.

[59] The Object Management Group.
http://www.omg.org.

[60] Objectivity - distributed object databases for high performance C++, Java and Smalltalk applications.
http://www.objectivity.com/.

[61] OptorSim homepage.
http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html.

[62] The Particle Physics DataGrid homepage.
http://www.ppdg.net.

[63] The PASTA reports.
http://david.home.cern.ch/david/pasta/pasta2002Report.htm.

[64] Private Communications - IBM Response to the University of Edinburgh and the University of Glasgow for the Establishment of Scotgrid.

[65] Prompt Reconstruction Operations Home Page.
http://www.slac.stanford.edu/BFROOT/www/Computing/Online-/PromptReco/index.html.

[66] The RAL BaBar Computing Farm.
http://hepunx.rl.ac.uk/BaBar/csflnx.html.

[67] Replica Optimization Service.
http://edg-wp2.web.cern.ch/edg-wp2/optimization/ros.html.

[68] The RooFit Toolkit for Data Modelling.
http://roofit.sourceforge.net.

[69] ROOT An Object-Oriented Data Analysis Framework. http://root.cern.ch/.

[70] The ScotGrid homepage.
http://www.scotgrid.ac.uk.

[71] The Seti@Home project.
http://setiathome.ssl.berkeley.edu/index.html.

[72] Simulation development group.
http://www.slac.stanford.edu/BFROOT/www/Computing/Offline-/Simulation/web/index.html.

[73] Skims and Streams in BaBar.
http://www.slac.stanford.edu/BFROOT/www/Physics/skims/skims.html.

[74] SOAP Version 1.2.
http://www.w3.org/TR/soap12-part1/.

[75] The SRB Homepage.
http://www.npaci.edu/srb/.

[76] stackthru (V0.99/R005).
http://www.scheibli.com/v3/projects/stackthru/.

[77] Threaded i/o tester.
http://sourceforge.net/projects/tiobench/.

[78] Top 500 Supercomputer Sites.
http://www.top500.org/.

[79] The UK Tier 1/A Centre.
http://www.gridpp.ac.uk/tier1a/.

[80] The Unicore project homepage.
http://www.unicore.org/.

[81] Web Services architecture overview.
http://www-106.ibm.com/developerworks/web/library/wovr/?dwzone=web.

[82] Web Services Resource Framework.
http://www.globus.org/wsrf/.

[83] What is MONO?
http://www.mono-project.com/about/index.html.

[84] Workbook for BaBar Offline Users - Objectivity: Reading the Object Database.
http://www.slac.stanford.edu/BFROOT/www/doc/workbook-/objectivity1/objectivity1.html.

[85] WP1 Workload Management homepage.
http://server11.infn.it/workload-grid/.

[86] WP10 Bio-informatics homepage.
http://edg-wp10.healthgrid.org/.

[87] WP12 Project Management homepage.
http://eu-datagrid.web.cern.ch/eu-datagrid/WP12/default.htm.

[88] WP2 Data Management homepage.
http://edg-wp2.web.cern.ch/edg-wp2/.

[89] WP2 Replica Management task.
http://edg-wp2.web.cern.ch/edg-wp2/replication/index.html.

[90] WP3 Information and Monitoring Services homepage.
http://hepunx.rl.ac.uk/edg/wp3/.

[91] WP4 Fabric Management homepage.
http://hep-proj-grid-fabric.web.cern.ch/hep-proj-grid-fabric/.

[92] WP5 Mass Storage Management homepage.
http://web01.esc.rl.ac.uk/projects/DataGrid/wp5/.

[93] WP6 Testbed homepage.
http://marianne.in2p3.fr/.

[94] WP7 Network Services homepage.
http://ccwp7.in2p3.fr/.

[95] WP8 High Energy Physics homepage.
http://datagrid-wp8.web.cern.ch/DataGrid-WP8/.

[96] WP9 Earth Observation Science Applications homepage.
http://styx.esrin.esa.it/grid/.

[97] Paul Baron. On distributed communications.
http://www.rand.org/publications/RM/RM3420/.

[98] T.A. Barrass, O.J.E. Maroney, S. Metson, and D. Newbold. Integrating the srb with the giggle framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 2004. http://www.sciencedirect.com/science/article-/B6TJM-4D07THF-F/2/086ea3cfb191596e38bd72963980c8b7.

[99] Fran Berman, Geoffrey Fox, and Anthony Hey, editors. *Grid Computing - Making the Global Infrastructure a Reality*, chapter 6, pages 171–198. Wiley, 2003.

[100] Fran Berman, Geoffrey Fox, and Anthony Hey, editors. *Grid Computing - Making the Global Infrastructure a Reality*, chapter 8, pages 217–250. Wiley, 2003.

[101] Fran Berman, Geoffrey Fox, and Anthony Hey, editors. *Grid Computing - Making the Global Infrastructure a Reality*, chapter 29, pages 701–712. Wiley, 2003.

[102] Fran Berman, Geoffrey Fox, and Anthony Hey, editors. *Grid Computing - Making the Global Infrastructure a Reality*, chapter 39, pages 859–905. Wiley, 2003.

[103] Fran Berman, Geoffrey Fox, and Anthony Hey, editors. *Grid Computing - Making the Global Infrastructure a Reality*, chapter 18, pages 471–490. Wiley, 2003.

[104] Fran Berman, Geoffrey Fox, and Anthony Hey, editors. *Grid Computing - Making the Global Infrastructure a Reality*, chapter 4, pages 101–116. Wiley, 2003.

[105] Charles E. Catlett and Larry Smarr. Metacomputing. *Communications of the ACM*, 35(6):44–52, June 1992.
http://portal.acm.org/citation.cfm?id=129890&coll=portal-&dl=ACM&CFID=15044097&CFTOKEN=59513139.

[106] Online Computer Library Center. Introduction to Dewey Decimal Classification.
http://www.oclc.org/dewey/versions/ddc22print/intro.pdf.

[107] Randy Chow and Theodore Johnson. *Distributed Operating Systems and Algorithms*, chapter 2. Addison Wesley, 1997.

[108] Randy Chow and Theodore Johnson. *Distributed Operating Systems and Algorithms*, chapter 6. Addison Wesley, 1997.

[109] Randy Chow and Theodore Johnson. *Distributed Operating Systems and Algorithms*, chapter 12. Addison Wesley, 1997.

[110] Randy Chow and Theodore Johnson. *Distributed Operating Systems and Algorithms*, chapter 8. Addison Wesley, 1997.

[111] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*, chapter 1, pages 1–28. Addison Wesley, 3rd edition, 2001.

[112] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*, chapter 2, pages 29–64. Addison Wesley, 3rd edition, 2001.

[113] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*. Addison Wesley, 3rd edition, 2001.

[114] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*, chapter 8, pages 309–352. Addison Wesley, 3rd edition, 2001.

[115] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*, chapter 14, pages 553–606. Addison Wesley, 3rd edition, 2001.

[116] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*, chapter 9, pages 353–384. Addison Wesley, 3rd edition, 2001.

[117] Joel M. Crichlow. *The Essence of Distributed Systems*, chapter 2. Prentice Hall, 2000.

[118] Joel M. Crichlow. *The Essence of Distributed Systems*, chapter 4. Prentice Hall, 2000.

[119] G. Drinkwater and S. Sufi. The CCLRC Data Portal. http://epubs.cclrc.ac.uk/work-details?w=30290.

[120] A. Earl, A. Khan, A. Hasan, and D. Boutigany. Private communications on BdbServer++.

[121] A. Earl, S. Thorn, and P. Clark. ScotGrid: A prototype Tier 2 Centre. Computing for High Energy and Nuclear Physics, September 2004.

[122] Alasdair Earl. SDSC Storage Resource Broker and Related Software Report. Technical report, Rutherford Appleton Laboratory, 2001. http://www.ph.ed.ac.uk/~aearl/SRB/ral_srb_reportv1.0.doc.

[123] Alasdair Earl, Nick Brook, Ian Stoke-Rees, and Andreas Pfeiffer. Private communications on ROOT and LCG.

[124] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 2001.
http://www.globus.org/research/papers/anatomy.pdf.

[125] Ian Foster. GlobusWorld State of the Union Address. 2004.
http://www.globusworld.org/program/slides/k4.pdf.

[126] Ian Foster and Carl Kesselman. Globus : A Metacomputing Infrastructure Toolkit.
ftp://ftp.globus.org/pub/globus/papers/globus.pdf.

[127] Ian Foster and Carl Kesselman, editors. *The Grid : Blueprint for a New Computing Infrastructure*, chapter 5, pages 105–130. Morgan Kaufmann, 1999.

[128] Ian Foster and Carl Kesselman, editors. *The Grid : Blueprint for a New Computing Infrastructure*, chapter 11, pages 259–277. Morgan Kaufmann, 1999.

[129] Ian Foster and Carl Kesselman, editors. *The Grid 2 : Blueprint for a New Computing Infrastructure*, chapter 22, pages 391–430. Morgan Kaufmann, 2004.

[130] Ian Foster and Carl Kesselman, editors. *The Grid 2 : Blueprint for a New Computing Infrastructure*, chapter 29, pages 593–622. Morgan Kaufmann, 2004.

[131] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
http://www.ietf.org/rfc/rfc3280.txt.

[132] Dr. Swamy N. Kandadai. Running HPL Linpack on IBM xSeries Linux Clusters. Technical report, IBM Corporation, 2003.
http://www.redbooks.ibm.com/redpapers/pdfs/redp3722.pdf.

[133] Jarek Nabrzyski, Jennifer Schopf, and Jan Weglarz, editors. *Grid Resource Management: State of the Art and Future Trends*, chapter 5, pages 53–69. Kulwer Academic Publishers, 2003.

[134] Jarek Nabrzyski, Jennifer Schopf, and Jan Weglarz, editors. *Grid Resource Management: State of the Art and Future Trends*, chapter 21, pages 341–357. Kulwer Academic Publishers, 2003.

[135] Jarek Nabrzyski, Jennifer Schopf, and Jan Weglarz, editors. *Grid Resource Management: State of the Art and Future Trends*, chapter 20, pages 321–340. Kulwer Academic Publishers, 2003.

[136] Jeffrey Richter. *Applied Microsoft .Net Framework Programming.* Microsoft Press, 2002.

[137] M. Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella. International Conference on Peer-to-Peer Computing, 2001.
http://www.globus.org/research/papers/gnutella.01.pdf.

[138] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. In *IEEE Internet Computing*, volume 6(1), 2002.
http://www.globus.org/research/papers/gnutella.iptps.pdf.

[139] Thomas Sandholm and Jarek Gawor. *Globus Toolkit 3 Core White Paper.* Globus, July 2003.
http://www-unix.globus.org/toolkit/3.0/ogsa/docs/gt3_core.pdf.

[140] Karl Solchenback. Promoting Grid Computing in Europe.
http://www.unicore.org/press/EU5_Karl_Solchenbach_ATL.pdf.

[141] Dimitrios Tsirigkas. Experience with LCG-2 and Storage Resource Management Middleware. Master's thesis, University of Edinburgh, 2004.

[142] S. Tuecke, K. Czajkowski, I. Foster, and al. Open grid services infrastructure (ogsi) version 1.0. Technical report, 2003.
http://www.globus.org/research/papers/Final-_OGSI_Specification_V1.0.pdf.

Index

Glossary

.Net Microsoft Web Services Framework

ACL Access Control List

ADS Atlas Data Store

ALICE A Large Ion Collider Experiment - see also Large Hadron Collider

AliEn Alice Environment

ANL Argonne National Laboratory, Illinois

API Application Programming Interface

ARDA A Realisation of Distributed Analysis for LHC

ARDA Advanced Research and Development Activity. A US funded agency for conducting advanced research and development related to information technology.

ARPA Advanced Research Programmes Agency. The initial funding agency in the United States for the internet with the ARPAnet. Currently known as DARPA.

ATLAS A TorolidaL ApparatuS - see also Large Hadron Collider

BaBar Collaboration The BaBar Collaboration consists of approximately 600 physicists and engineers from 75 institutions in 10 countries. They use the BaBar detector, which was built at SLAC, to study the millions of B mesons produced by the PEP-II storage ring

BaBarGrid BaBarGrid is a collection of developers who are developing the Grid capabilities of the BaBar experiment in terms of data transfer, simulation production and data analysis.

BBCP BaBar Copy. An extension of the Unix cp program.

BBSRC Biotechnology and Biological Sciences Research Council

Belle A B-Factory experiment based at KEK, Japan

BLOB Binary Large Object

BNL Brookhaven National Laboratory, New York

CA Certificate Authority

CASPUR Consorzio interuniversitario per le Applicazioni di Supercalcolo Per Universitá e Ricerca

CASTOR Cern Advanced STORage Manager

CCLRC Council for the Central Laboratories of the Research Councils

CE The Compute Element is the interface to the processing capabilities of the local system. It maintains the scheduler, or schedulers, which allow users to submit interactive and batch requests.

CERN European Laboratory for Particle Physics

CM0 The computing model used for the BaBar experiment prior to the start of data taking and for the 1999 and 2000 production runs.

CM1 The second BaBar computing model. A transitional model between CM0, which used Objectivity, and CM2, which uses KANGA.

CM2 The third computing model used for the BaBar experiment. In this model the use of the Objectivity database management system was phased out in favour of a KANGA based system for the event store and conditions database.

CMS Compact Muon Solenoid - see also Large Hadron Collider

CNRS Centre national de la reserche scientifique - France

COBRA Common Object Request Brokering Architecture

cyberinfrastructure US nomenclature for e-Science

DAI Data Access and Integration

DAIT Data Access and Integration Two - successor to DAI

Data Challenge A test of an experiments computing resources which is used to generate a specified quantity of simulation data in a finite time.

Data management Work that involves the planning, development, implementation, and administration of systems for the acquisition, storage, and retrieval of data.

datagrid A datagrid is a Grid which has been developed with a primary focus on data problems. This includes: grids which deal with a very large volume of data, such as the European DataGrid; ones where data is time critical, requiring low latency, such as command and control systems; and ones where data transfer must be reliable and robust, such as banking networks

dCache A system for storing and retrieving data, distributed among a large number of heterogeneous server nodes, under a single virtual filesystem tree with a variety of standard access methods. dCache was designed and implemented at DESY.

DCAP dCache Access Protocol

DCOM Microsoft Distributed Component Object Model

DESY Deutsches Elektronen-Synchrotron

DICE Data Intensive Computing Environments - US project

digi The smallest data element provided by the BaBar experiment.

DIRAC Distributed Infrastructure with Remote Agent Control

DL Daresbury Laboratory - see also CCLRC

DOD United States Department of Defence

DOE United States Department of Energy

DSOM IBM Distributed System Object Model

DTi Department of Trade and Industry

duplex mode A full-duplex channel is a network channel which can carry information in both directions at once.

Durable Space SRM - storage space where files have lifetimes but where the system cannot delete them automatically if their lifetime is expired if they are still referenced by the client.

e-Science The term e-Science was coined in 1999 by John Taylor, the Director General of the United Kingdom's Office of Science and Technology.

> e-Science refers to the large scale science that will increasingly be carried out through distributed global collaborations enabled by the Internet. Typically, a feature of such collaborative scientific enterprises is that they will require access to very large data collections, very large scale computing resources and high performance visualisation back to the individual user scientists.

[DTi def.]

EDG European DataGrid

EDI Electronic Data Interchange

eDikt eScience Data Information and Knowledge Transformation

EGEE European Grid for Enabling EScience

Enterprise Grid A Grid which is built from various Local Grids within the same organisation to provide facilities on-the-fly.

EPCC Edinburgh Parallel Computing Centre

EPSRC Engineering and Physical Sciences Research Council

ESA European Space Agency

ESRC Economic and Social Research Council

Event Store (BaBar) The Event Store manages and tracks the experimental data from the initial raw data produced either by experiment or simulation through various reconstruction and physics analysis processing phases. It also provides a mechanism for creating specific sets of events.

Factory In Grid terms, a secure and stateful persistent service. This is used to invoke stateless web services.

FNAL Fermi National Accelerator Laboratory, Illinois

GANGA Gaudi / Athena and Grid Alliance. A joint ATLAS and LHCb Grid interface.

GASS Grid Access to Secondary Storage

GEANT A multi-Gigabit pan-European data communications network for research and education

GEANT GEANT is a system of detector description and simulation tools.

GFAL Grid File Access Layer

GFAL Grid File Access Library

GGF Global Grid Forum

GIIS Grid Index Information Service

Global Grid An Grid system with resources and information potentially being available for multiple Enterprise Grids as well as application service providers, utility computing centres and user provided services.

Globus The Globus Project develops the Globus Toolkit for enabling the constructions of Grids

GPFS Grid Parallel File System

GPT Grid Packaging Toolkit

GRAM Globus Resource Allocation Manager

Grid A three tier distributed computing architecture where standards based middleware provides an abstract layer between service providers and client application.

Grid Computing Provides pervasive, dependable, consistent and inexpensive access to advanced computational capabilities, databases, sensors and people.

Grid Services A Web Service which fulfils the OGSI requirements.

GRIS Grid Resource Information Service

GSH Grid Service Handle

GSR Grid Service Reference

GT2 Globus Toolkit Version 2.x. A Unix based Grid Toolkit.

GT3 Globus Toolkit Version 3.x. A OGSA based Grid Toolkit.

GT4 Globus Toolkit Version 4.x. A WS-RF based Grid Toolkit.

GUID Grid Unique Identifier

GWSDL Grid Web Services Description Language. An extension to WSDL Version 1.2

half-duplex mode A half-duplex channel is one that can carry information in both directions, but not at the same time e.g. one will tramsmit while the other listens for a fixed period before switching roles.

HEFCE Higher Education Funding Council for England

HERA Hadron-Electron Ring Accelerator

HPDC High Performance and Distributed Computing

HPSS High Performance Storage System

HSM Hierarchical Storage Manager

IETF Internet Engineering Task Force

IN2P3 Institut National de Physique Nucléaire et de Physique des Particules

INFN Istituto Nazionale di Fisica Nucleare

Information Grid A Grid which has been developed specifically to return information, or meta data, to the user. Information Grids concentrate on ensuring that there is processing capacity very close to where the raw data is stored so that remote processing can occur and the minimum set of data can be returned to the user.

IP Internet Protocol

J2EE Java 2 Enterprise Edition

JDL Job Description Language

JLab Thomas Jefferson National Accelerator Facility, Virginia

KANGA Kinder ANd Gentle Analysis. Data format for BaBar.

LBNL Lawrence Berkeley National Laboratory

LCFG server The Local ConFiGuration system was developed in the School of Informatics at the University of Edinburgh and has been adopted by LCG to configure local systems. The LCFG server maintains an XML description of each system in the local network which allows system administrators to install and maintain their systems using RPMs.

LFN Logical Filename

LHC The Large Hadron Collider, based at CERN and due to commence taking data in 2007.

LHCb The LHC beauty experiment - see also Large Hadron Collider

LIGO Laser Interferometer Gravitational-Wave Observatory

Local Grid A Grid which is limited to a local group or department.

LTO Linear Tape-Open

MCAT Metadata CATalogue - see also SRB

MDS Metadata Directory Service

Media driver In SRB a piece of software which provides instructions to a hardware resource based on a standard interface

Micro (BaBar) The Micro database is intended to be used mainly as a quick way of doing analysis.

Middleware Software which is above the resource / operating system layer but below the application layer. Commonly used for tasks such as distributed security, resource location, etc.

Mini (BaBar) A BaBar Event data format that stores reconstructed detector objects (hits, tracks, clusters, ...) in a compact form.

MRC Medical Research Council

MTBF Mean Time Before Failure

NAFS Network Attached File System

NAS Network Attached Storage

NCSA National Center for Supercomputing Applications, Illinois

Near-line storage Several types of robots for magnetic tape and cassettes exist to make off-line storage available all the time (near-line) un-operated, un-assisted with a delay of a few seconds.

NERC Natural Environment Research Council

NFS Networked File System

NIKHEF National Institute for Nuclear Physics and High Energy Physics - the Netherlands

NPACI National Partnership for Advanced Computer Infrastructure

NSF National Science Foundation

OASIS Organisation for the Advancement of Structured Information Standards

ODBMS Object Database Management System

OEP Online Event Processing (BaBar)

OGSA Open Grid Services Architecture

OGSI Open Grid Services Infrastructure

OMG Object Management Group

P2P Peer-to-Peer. A non-hierarchical network of computer systems where all nodes have the ability to locate, communicate, and route data to, other nodes.

PEP-II An asymmetric B Factory located at SLAC which provides B mesons for the BaBar detector.

Permanent Space SRM - storage space where files can be archived and can only be deleted by the owner.

PNFS Perfectly Normal File System

PPARC Particle Physics and Astronomy Research Council

Primary online storage Random access storage directly connected to a processing unit, the systems working store or main memory, typically RAM or ROM.

Proxy certificate A certificate with a short lifetime, which is generated by a user from their Grid Certificate and private key. This allows restricted access to resources they are authorised to use.

Pull Model A computing model where a node or resource requests data and information from nodes beneath it in a hierarchical structure.

Push Model A computing model where resources publish, or push, information and data to other resources which then aggregate it.

R-GMA Relational Grid Monitoring Architecture

RAID Redundant Array of Inexpensive Disks

RAL Rutherford Appleton Laboratory - see also CCLRC

RAM Random Access Memory

Raw (BaBar) The Raw database contains the Raw digis out of the detector.

RB Resource Broker

Rec (BaBar) Reconstructed data.

RFIO Remote File Input/Output

RLS Replica Location Service

RMI Sun Microsystems Java Remote Method Invocation

ROM Read Only Memory

RPC Remote Procedure Call

RPM RedHat Package Manager - a software installation and maintenance system

RRS Replication and Registration Services

SAML Security Assertion Markup Language

SAN Storage Area Network

Sand-box A restricted environment, typically in memory, where untrusted software can be run without the ability to affect stored data or the system.

ScotGrid Scottish Tier 2 centre for LHC computing

SDK Software Development Kit

SDSC San Diego Supercomputing Center, California

SDSS Sloan Digital Sky Survey

Secondary online storage Fast non-volatile storage, in practise often local magnetic disks, special disk configurations (i.e. RAID), tape which is cached on disk, and non-local network storage.

SFN Site Filename. See SURL

SHEFC Scottish Higher Education Funding Council

single sign-on A system where users are required to authenticate themselves once, after which they can use the resources in the system without re-authentication.

SLAC Stanford Linear Accelerator Centre, California

SNIA Scalable Node Intel Architecture (SGI)

SOAP Simple Object Access Protocol

Soft State A system which assumes unreliable connections and nodes and therefore checks node availability and connection rather than using prior information.

Spitfire A Grid interface to relational databases developed by Work Package 2 of the European DataGrid.

SRB Storage Resource Broker - hierarchical storage system from SDSC

SRB ticket In SRB a ticket is a 10 digit token which allows restricted access to resources and/or data

SRIF Science Research Investment Fund

SRM Storage Resource Manager

SRM2SRB The SRM interface to SRB project

SSH Secure SHell. A Unix shell interface with enhanced security commonly used for accessing remote systems

SSL Secure Socket Layer - see also TLS

StFN Storage Filename

Storage Element The Storage Element (SE) provides remote clients with an interface to data stored on the local system and as a retrieval agent for the local Compute Element and Worker Nodes. It interacts with the Replica Location Service (RLS) to register local data, to update the RLS about modifications to this data, and to locate data for the WNs.

SURL Site filename as URL. See SFN

Tag (BaBar) This is a highly compressed summary that classifies the event and allows simple selection queries to be performed without accessing other information.

TDR Technical Design Report

Tertiary offline storage Tapes and other such media used for data backup, usually either : near-line, automated operation (tape robots); offline, manually operated; or tertiary long term, archival and backup storage such as CD-R.

Tier 1 Centre In the LHC Computing model a Tier 1 Centre is a national computing and data storage facility with a Memorandum of Understanding on agreed service levels, resources and staffing.

Tier 2 Centre In the LHC Computing model a Tier 2 Centre is a regional computing centre which services various institutes and groups but with less resources than a Tier 1 Centre and no MoU.

Tier 3 Centre In the LHC Computing model a Tier 3 Centre is an institute or group which analyses data from the LHC experiments.

Tier A In the BaBar Computing Model a national computing centre similar to an LHC Tier 1 Centre.

Tier B In the first BaBar Computing Model (CM1) a regional computing centre. This was depreciated with the development of the second computing model, CM2.

Tier C In the BaBar Computing Model an institute or group which makes use of the data generated by the experiment.

TLS Transport Layer Security. A replacement for SSL.

TURL Transfer URL. See SRM and URL

UDDI Uniform Description, Discovery and Integration

Unicore Uniform Interface to Computing Resources

URI Uniform Resource Identifier. Consists of a transfer type (such as ftp, file, http) followed by a colon followed by a path.

URL Uniform Resource Locater. Global address of a file or system. See URI

User Interface The User Interface (UI) is the job submission tool of LCG. It can be installed in the user's home directory and does not require the same installation complexity of the other LCG resource management tools.

VO Virtual Organisations are groups of individuals and resources spread between physical locations and administrative domains. They address a common aim by the sharing of computing and data resources and specialist equipment.

Volatile Space SRM - storage space where files have lifetimes associated with them. The system can reclaim space by deleting files once the files lifetime is exceeded.

VOMS Virtual Organisation Membership Service

W3C World Wide Web Consortium

Web Services Web services are online services whose public interfaces and bindings are defined and described using XML

Work-flow The sequence of actions, methods or functions which are required to achieve a specific aim. This can require multiple software packages and computer systems.

Worker Node(s) WNs provide the computing capabilities of the local system. They are managed by the CE and can interact with the other local systems. Significant debate has gone into the issue of whether they should be directly connected to the Internet for communications or not.

WSDL Web Services Description Language

WSFL Web Services Flow Language

WSRF Web Services Resource Framework

xinetd Extended Internet Services Daemon

XKMS XML Key Management Specification

XML eXtensible Markup Language