ESTBLISHMENT OF GRID COMPUTING ENVIRONMENT FOR GRIDFTP CLIENT SERVER MODEL

A Dissertation

Submitted in partial fulfillment

of the requirement for the award of the Degree of

MASTER OF ENGINEERING
in
COMPUTER TECHNOLOGY & APPLICATIONS

By

MANU AGARWAL
College Roll No. (03/CTA/03)
Delhi University Roll No. 3003

Under the guidance of

Dr. Goldie Gabrani

Department of Computer Engineering
Delhi College of Engineering,
University of Delhi

2003-2005

CERTIFICATE

This is to certify that the work that is being presented in this dissertation entitled ***"Establishment of Grid Computing Environment for Grid FTP Client Server model"*** submitted by **Manu Agarwal** in the partial fulfillment of the requirement for the award of the degree of **Master of Engineering** in Computer Technology and Application, Delhi College of Engineering is an account of his work carried out under my guidance and supervision.

The work embodies in this dissertation has not been submitted for the award of any other degree to the best of my knowledge.

Professor D. Roy Choudhury
Head of Department
Department of Computer Engineering
Delhi College of Engineering
Delhi

Dr. Goldie Gabrani
Asst. Professor
Department of Computer Engineering
Delhi College of Engineering
Delhi

ACKNOWLEDGEMENT

It is a great pleasure to have the opportunity to extent my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor **Dr. Goldie Gabrani**, Assistant Professor in the Department of Computer Engineering, Delhi College of Engineering, for her invaluable guidance, encouragement and patient review. His continuous inspiration only has made me complete this major project.

I would also like to take this opportunity to present my sincere regards to my teachers viz. Professor D. Roy Choudhury, Dr S. K. Saxena, Mr. Rajeev Kumar and Mrs. Rajni Jindal for their support and encouragement.

I would like to express my heartiest felt regards to Professor Asok De, Head of Computer Center for providing Linux lab for my work.

I am thankful to my friends and classmates for their unconditional support and motivation during this project.

Manu Agarwal
M.E. (Computer Technology & Applications)
College Roll No. 03/CTA/03
Delhi University Roll No. 3003

ABSTRACT

A Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities. “Grid” computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications and high-performance orientation.

The aim of this dissertation is to establish a grid computing environment for GridFTP client server model, a data management component using Globus toolkit 2.4. The Grid toolkit is widely used tool and libraries for grid computing including certificate based authentication and comprises a set of components that implement basic services for security, resource location, resource management, data management, communication etc. We present an extensible Grid architecture, in which protocols, services, application programming interfaces, and software development kits are categorized according to their roles. We describe GridFTP service extends the popular FTP protocol with new features required for Grid applications, such as striping and partial file access. Grid FTP C API is used to implement different functionality of protocol.

CONTENTS

Chapter 1

INTRODUCTION

Distributed computing [1] is a subject that has been studied a lot in the past, but recently a new field, Grid computing, has drawn attention back to the area. Grid computing differs from traditional distributed computing in its goals: sharing resources on heterogeneous platforms on a geographically wide area is now coming possible through advances in network technology.

A grid [2, 36] enables the sharing, selection, and aggregation of a wide variety of geographically distributed resources including supercomputers, storage systems data sources and specialized devices owned by different organizations administrated with different policies. Grids are typically used for solving large-scale resource and computing intensive problems in science, engineering, and commerce.

Grid computing [3, 35] is a sub set of distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of the connected heterogeneous systems sharing various combinations of resources. The standardization of communications between heterogeneous systems created the Internet explosion. The emerging standardization for sharing resources, along with the availability of higher bandwidth, are driving a possibly equally large evolutionary step in grid computing. Grid computing involves coordination and networking of resources across dynamic and geographically dispersed organizations in a transparent way for users. Grid technologies emphasize effective operation in large scale, wide area environments, including access to remote computation, information services, high speed data transfers and gateways to local authentication schemes.

1.1 Literature Survey

Distributed computing has fascinated researchers all over the world for past two decades. Traditionally the focus has been on developing a unified homogeneous distributed system.

Recently, there has been tremendous growth in wide area distributed computing leading to grid computing.

The term Grid is chosen as an analogy to electric power grid that provides consistent, pervasive, dependable, transparent access to electricity, irrespective of type and location of source. The primary focus in Grid computing is to harness the power of geographically distant supercomputers, and convert them into a big computing resource.

The Grid enables sharing, selection and aggregation of various resources including raw CPU cycles, storage systems, data sources and special services like application servers, etc. These resources may be geographically dispersed, operated by different organizations with different policies running on completely different operating systems. Figure 1.1 is the simplest grid consists of just a few machines, all of the same hardware architecture and same operating system, connected to local network.

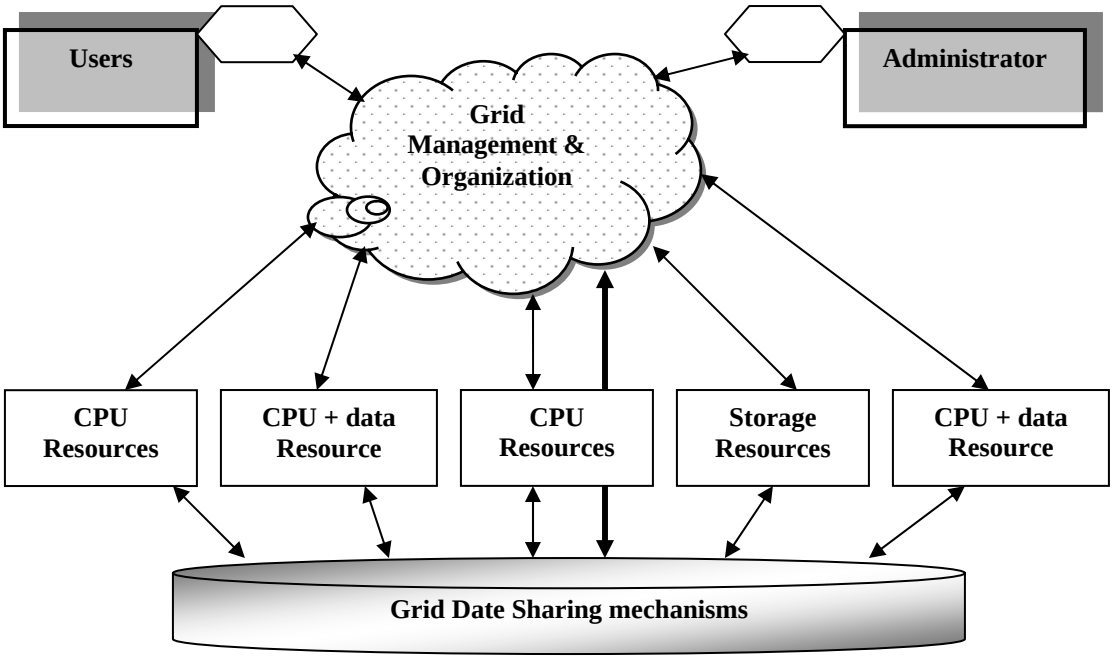

Figure 1.1: The Simple Grid

This kind of grid uses homogeneous systems so there are fewer considerations and may be used just for experimenting with grid software. The machines are usually in one department of an organization, and their use as a grid may not require any special policies or security concerns. Because the machines have the same architecture and operating system, choosing application software for these machines is usually simple. Some people would call this a "cluster implementation rather than a "grid."

Due to the recent explosion of Grid Technologies, there has been some confusion over the exact nature of a Grid. Dr. Foster provides a three point checklist [4] for evaluating grid systems.

A grid is a system that

- Coordinates resources that are not subject to centralized control - A Grid integrates and coordinates resources and users that live within different control domains for example, the user's desktop vs. central computing; different administrative unit of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system.

- Uses standard, open, general-purpose protocols and interfaces - A Grid is built from multi-purpose protocols and interfaces that address such fundamental issue as authentication, authorization, resource discovery, and resource access. It is important that these protocols and interfaces be standard and open. Otherwise, we are dealing with an application-specific system.

- Delivers nontrivial qualities of service - A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co- allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts.

Grids are composed of VO (Virtual Organizations), a conglomeration of network resources consisting of servers, desktop PCs, mainframes, clusters, etc. These VOs are managed by different organizations and may have different policies and security mechanisms. Grid enables sharing of resources between these heterogeneous VOs with a common set of open protocols. There are enormous opportunities for application writers to exploit the large amount of computational and storage resource provided by the grid. There are enormous opportunities for application writers to exploit the large amount of computational and storage resource provided by the grid.

1.1.1 Grid System Taxonomy

A grid should provide full-scale integration of heterogeneous computing resources of any type: processing units, storage units, communication units, and so on. However, as the technology hasn't yet reached its maturity, real-world grid implementations are more specialized and generally focus on the integration of certain types of resources. Different types of grids describe as follows [5]:

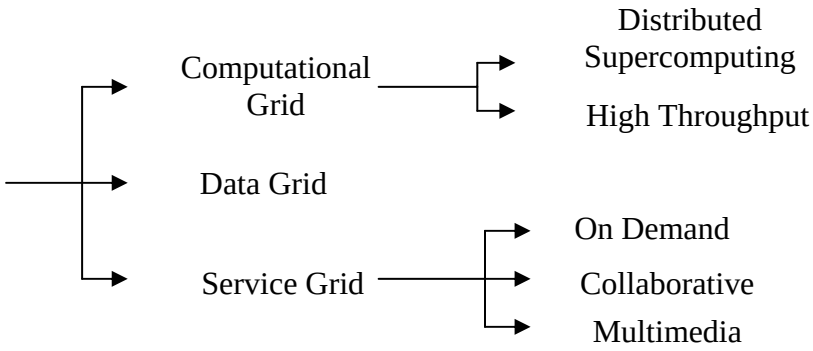

Figure 1.2: Grid Systems Taxonomy

1. Computational grid: A computational grid is a grid that has the processing power as the main computing resource shared among its nodes. This is the most common type of grid and it has been used to perform high-performance computing to tackle processing-demanding tasks.

2. Data grid: A data grid has the data storage capacity as its main shared resource. Such a grid can be regarded as a massive data storage system built up from portions of a large number of storage devices.

3. Service grid This is known as either a service grid or a delivery grid. Such a grid has as its main purpose to provide fault-tolerant and high-performance communication services. In this sense, each grid node works as a data router between two communication points, providing data-caching and other facilities to speed up the communications between such points. Examples of such grids are On Demand computing, Multimedia computing or Collaborative computing.

1.1.2 Characteristics of Grid and its application

A computational grid provides high-performance computing; a data rid provides large storage capacity; and a network grid provides high throughput communication that may be useful for a variety of applications, such as virtual conferences. So main reasons for using grid computing as follows:

- Improve efficiency/reduce costs
- Exploit under-utilized resources
- Enable collaborations
- Virtual resources and virtual organizations (VO)
- Increase capacity and productivity
- Parallel processing capacity
- Support heterogeneous systems
- Provide reliability/availability
- Access to additional resources
- Resource balancing
- Reduce time to results

Applications

Grid implementations only make sense in environments where a meaningful number of computing resources can be integrated to form a higher-performance system, which tends to be rather restrictive. Grid technology in different industries, but it is not restricted to only those areas. Grid computing capability is growing everyday, and we believe that someday, all

systems and applications will run in some kind of Grid like environment. The following list is far from complete, but represents potential for the present and future:

Finance

- Derivative analysis
- Statistics
- Portfolio risk
- Insurance policy cost
- Real-time stock market analysis

Life sciences

- Drug screening
- Protein folding
- Protein sequencing

Medicine

- Record management
- Automated analysis and diagnosis
- Research into the nature of disease

Government/Academia

- Dispersed research center collaboration
- Weather forecasting
- General high-performance computing

Energy

- Seismic analysis
- Oil field simulation

Manufacturing

- Product design
- Simulation
- Modeling
- Finite element analysis (anything involving flow (air, water, fuel, and so on)

Telecommunications/Media

- Video rendering
- Network gaming
- Content distribution
- Dynamic bandwidth for new classes of applications

Electronics

- Chip layout optimization
- Board layout optimization
- Circuit simulation

1.2 Grid basic Architecture

In this section we present Grid architecture as it is described in the Globus project [6]. Figure 1.2, Grid Architecture represents a conceptualization of the main principles and requirements in Grid environments. The motivation for building this architecture is the need for a new model describing sharing of heterogeneous resources. This architecture identifies the basic components of Grid systems, defines the purpose of such components, and finally indicates how these components interact with each other.

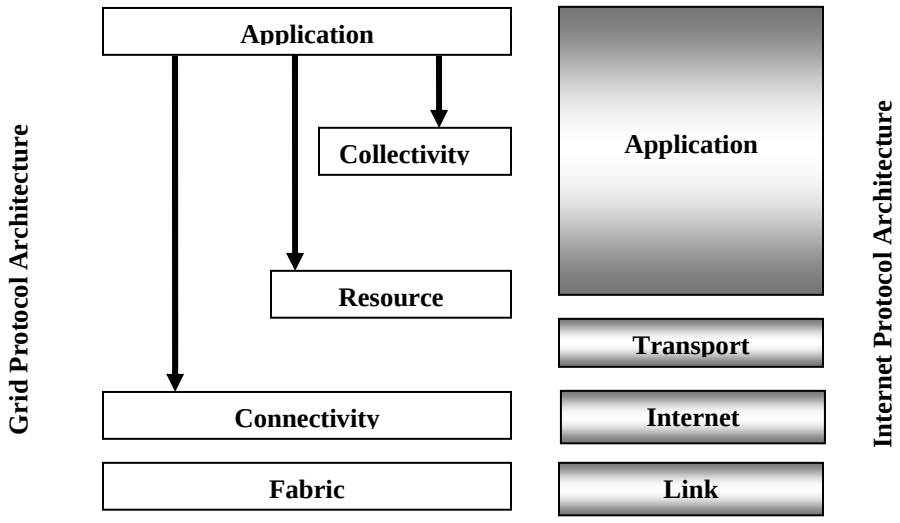

Figure 1.3: High-level Grid Architecture and Functional Blocks

The architecture of the Grid is described in terms of layers, each providing a specific function. In general, higher layers are focused on the user (user-centric), whereas lower layers are more focused on computers and networks (hardware-centric). The different layers and functionalities are described as follows:

- **Fabric layer:** Interfaces to local control, of physical and logical resources. The fabric layer is composed by computational resources, storage systems, catalogs, distributed file systems, network resources, and sensors to be share.

- **Connectivity layer:** Defines core communication and authentication protocols supporting Grid-specific network transactions.

The authentication protocols are governed by the following principles:

Single sign on applies to enabling the user to have multiple access to the resources from the Fabric layer during the same login, once the authenticity has been established. That is, once sign on is performed, the user is authenticated for the entire Grid.
Delegation designates the ability to provide a program with the appropriate rights such that it could behave on user's behalf and further access those resources to which the user has permissions.

Integration with various local security solutions addresses the issue of allowing communication with the local security solutions by providing mapping to the local environment. For instance, Grid security should be able to cooperate with Kerberos and Unix security which could be implemented by the providers of sites or resources.

User-based trust relationships are concerned with directing the security constrains from the user to the intended resources and not further among their providers.

- **Resource layer**: Allows the sharing of a single resource. This layer includes protocols for control and management of individual resources.

Two primary classes of resource layer protocols can be distinguished:

Information protocols are used to obtain information about the structure and state of a resource, for example, its configuration, current load, and usage policy.

Management protocols are used to negotiate access to a shared resource, specifying, for example, resource requirements (including advanced reservation and quality of service) and the operation(s) to be performed, such as process creation or data access.

Management protocols are responsible for instantiating sharing relationships, ensuring that the requested protocol operations are consistent with the policy under which the resource is to be shared. Issues that must be considered include accounting and payment. A protocol may also support monitoring the status of an operation and controlling the operation.

- **Collective layer**: Allows resources to be viewed as collections. This layer includes all the services that allow us to manage several resources.

Examples of services are:

Directory services enabling the discovery of resources. A directory service supports queries for resources by name or by attributes such as type, availability, or load. Monitoring and diagnostics services enabling fault detection, such as overload, failure, intrusion.

Grid-enabled programming systems thus extending their functionality by augmentation. Software discovery services envisaging the discovery and selection of the best software and platform for solving a selected problem.

- **Application layer:** Uses the appropriate components of each layer to support the application. Applications Grid access to the infrastructure. According to the requirements of the application, it can be necessary to happen through all the layers or to connect themselves directly to the infrastructure.

1.3 Statement of the Problem

The goal set for the work, which is being presented in this dissertation, can be stated as follows:

1. To establish the grid computing environment for the different services of grid.
2. To design and implement the some of the GridFTP functionality and analysis the performance in data transfer.

1.4 Organization of the Dissertation

The reminder of this Dissertation is organized as follows: Chapter two presents an overview of Globus Grid toolkit 2.4 and its architecture and description of different component of toolkit in detail. Chapter three explains the GridFTP with its characteristics and different functionality for transferring data from one system to other in grid environment. Chapter four describes the setup of grid along with systems requirements, installation consideration like security, information services etc. Chapter five presents the design and implementation of the GridFTP and its programming environment. And finally, chapter six lists our conclusions and suggests some areas for future works.

Chapter 2

Grid Computing in the Globus Environment

A middleware is software that organizes and integrates the disparate computational facilities belonging to a Grid. Its main role is to automate all the machine to machine (M2M) negotiations required to interlace the computing and storage resources and the network into a single.

All major Grid projects [37] are being built on protocols and services provided by the Globus Toolkit, a software "work-in-progress" which is being developed by the Globus Alliance. It provides a set of software tools to implement the basic services and capabilities required to construct a computational Grid, such as security, resource location, resource management, and communications. This is an open-source initiative to produce a standard Grid architecture for distributed resources. Initially, Globus was intended to provide a secure means to submit jobs to a third-party scheduling and clustering system. It is based on the premise that grid computing can be seen as three pyramids built on top of a security infrastructure [7]:

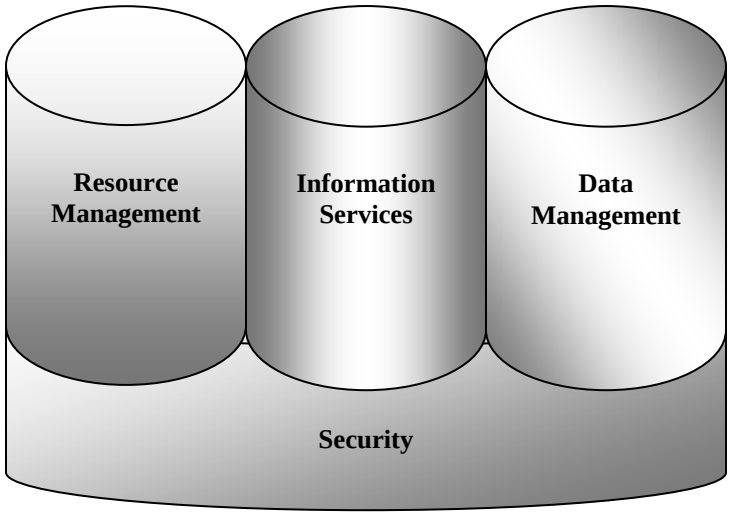

Figure 2.1: The Three pyramids

Examples of applications work being done by groups around the world include:

- **Smart instruments**: Advanced scientific instruments, such as, electron microscopes, particle accelerators, and wind tunnels, coupled with remote supercomputers, users, and databases, to enable interactive rather than batch use, online comparisons with previous runs, and collaborative data analysis.

- **Computationally enhanced desktops**: Software packages, such as, chemical modeling and symbolic algebra that transfer computationally intensive operations to more capable remote resources.

- **Collaborative engineering**: High-bandwidth access to shared virtual spaces that support interactive manipulation of shared data sets and steering of sophisticated simulations for collaborative design of complex systems.

- **Distributed computing**: Virtual supercomputers constructed from many individual supercomputers to solve problems too large for any single computer to accommodate.

- **Parameter studies**: Rapid, large-scale parametric studies, in which a single program is run many times in order to explore a multidimensional parameter space.

There are other software packages that provide similar services, but the Globus Toolkit differs from these in three significant ways:

- Its bag of services approach, which allows application software to use components of the Globus Toolkit without having to adopt the whole Globus Toolkit or a particular programming model or language.

- Its provision of specialized mechanisms that usually coexist with but also sometimes replace mechanisms provided by commodity computing.

- Its support for an information-based approach to meeting application performance requirements.

Using the basic services provided by the Globus Toolkit, researchers may build a range of higher-level capabilities. For example, the Globus Toolkit provides a complete implementation of the Message Passing Interface (MPI) that can run across heterogeneous collections of computers.

Due to its bag of services approach the Globus Toolkit can be used in different ways. The Globus Toolkit can be used at a site that wishes to participate in a computational grid to contribute resources to a grid's pool of resources. (The use of a site's grid resources is closely controlled by the site's access policies.) The Globus Toolkit can also be used to provide access to other grid resources without contributing any of a site's own resources that is, assuming that the appropriate access policies have been negotiated with the owners of the other resources. The Globus Toolkit also provides other services, like single sign-on authentication, without the need for contributing computational resources.

2.1 Globus Basic Service Architecture

Globus Toolkit 2.4 [8] provides some basic services that should be found in a grid. Each service needs a standard protocol or component because a grid deals with diverse types of resources.

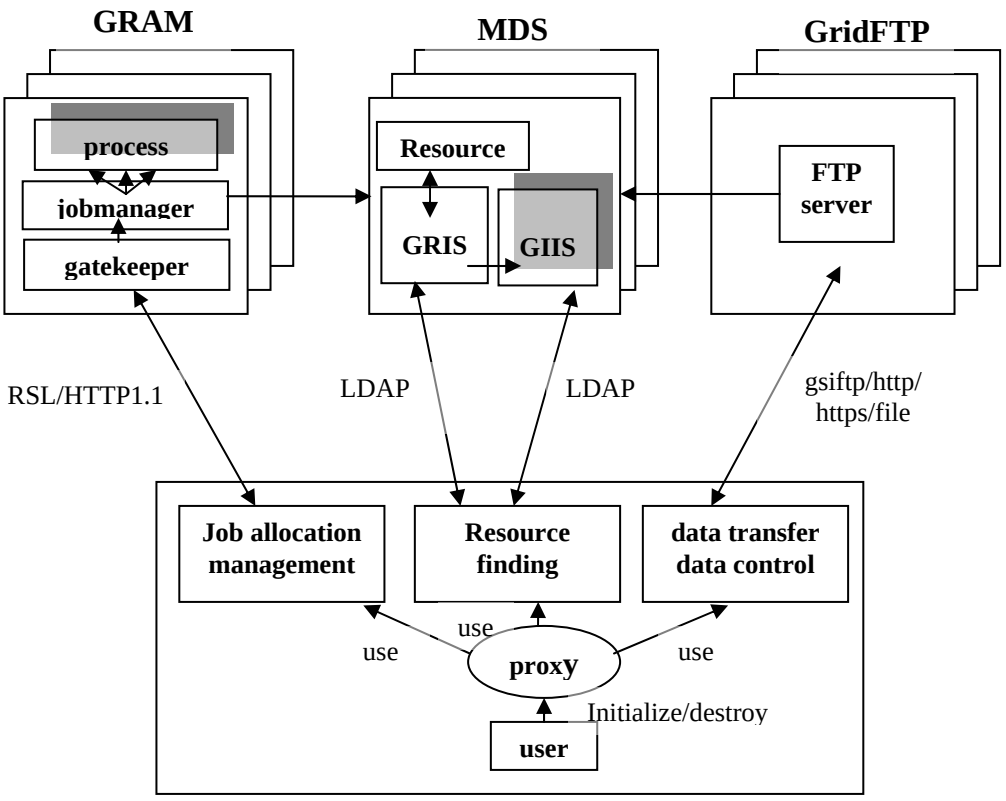

Figure 2.2: The system overview of Globus Toolkit

2.2 Security Service

Grid Computing, being distributed and heterogeneous in its nature, has high demands on security. Security service is implemented by the Globus Security Infrastructure (GSI) [9, 10]. To implement that, GSI utilizes Secure Socket Layer (SSL) [11] protocol, public key encryption [12], and X.509 [13] certificates.

The primary motivations behind the GSI are:

- The need for secure communication between elements of a computational Grid.
- The need to support security across organizational boundaries, thus avoiding a centrally managed security system.
- The need to support "single sign-on" for users of the Grid, including computations that involve multiple resources and/or sites.

The Certificate Authority is one of the most important aspects of maintaining strong grid security. An organization may choose to use an external Certificate Authority or operate one itself. Authority to strictly adhere to its responsibilities.
The primary responsibilities of a Certificate Authority are:

- Positively identify entities requesting certificates
- Issuing, removing, and archiving certificates
- Protecting the Certificate Authority server
- Maintaining a namespace of unique names for certificate owners
- Serve signed certificates to those needing to authenticate entities
- Logging activity

GSI gives access to the grid using a Certificate Authority and a set of keys for public key cryptography. The routine to establish the GSI communication starts with copying the CA's public key to the GSI client which generates a private key and a certificate request. The certificate is sent to the CA which signs it and sends it back to the GSI client. Then secure communication is established as the client possesses a private key, the public key of the CA and his (her) digital certificate.

Once a new grid host has successfully gained access to the grid, it can start communicating with other hosts. When setting up a communication between two hosts, the first task is to be able to determine which host is certified and which is not. The GSI authentication process is summarized in the following illustration (Figure 2.3).

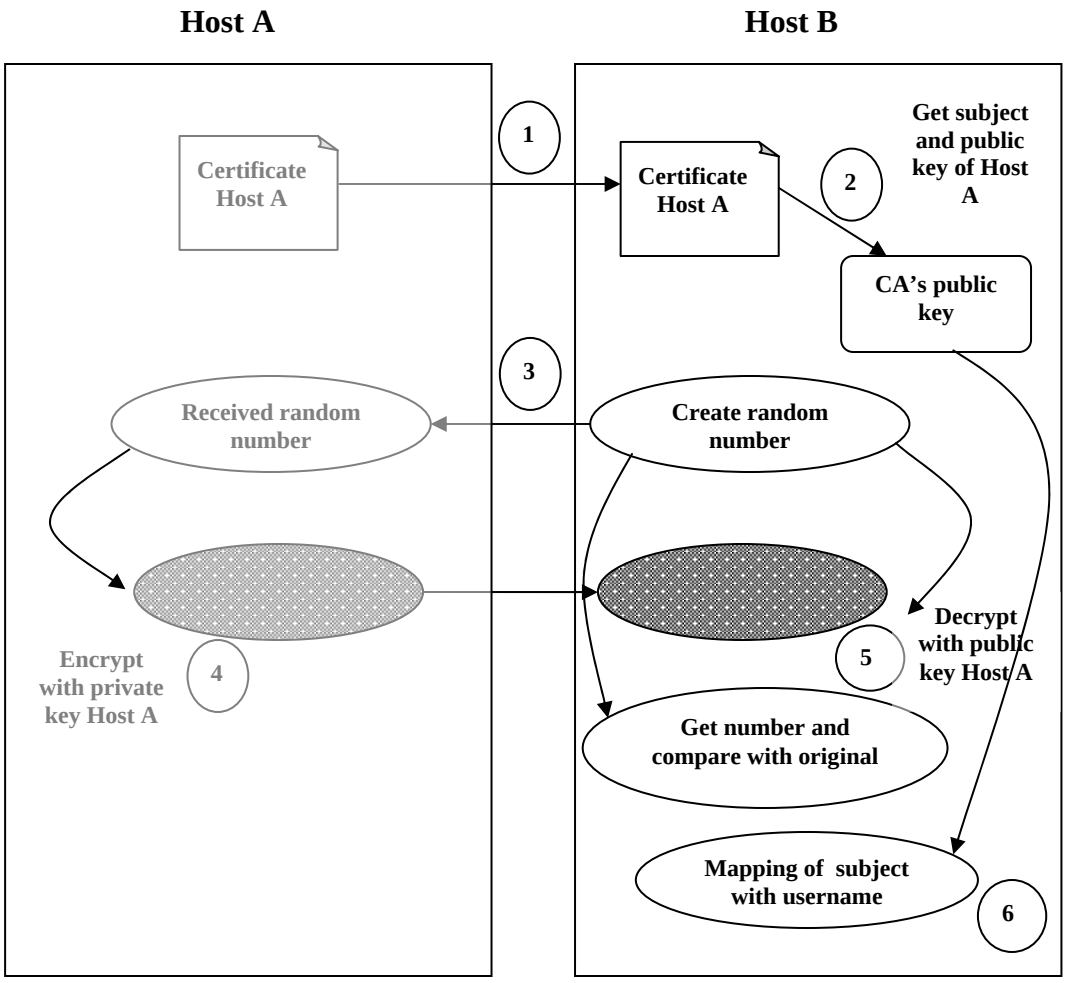

Figure 2.3: Authentication Procedure between two grid hosts

This short illustration shows that certificates and encryption keys (public or private) are the requested digital documents for grid authentication. The procedure consists in checking that the certificate and the keys of a given host are coherent.

Authorization mechanism: users that are willing to be authorized to use GSI-enable services need to belong to the GSI access control list. The GSI administrator verifies that the GSI Identity is owned by the username requesting the service.

2.3 Resource Management

The resource management [14] pyramid provides support for:

- Resource allocation
- Submitting jobs: Remotely running executable files and receiving results
- Managing job status and progress

Component of resource management are as follow:

- **GRAM**

Grid Resource Allocation Manager (GRAM) [15] Reports monitor and publishes information about the identity and state of local computations (registry). Moreover, it allows users to schedule and manage remote computations. Specifically, various classes and methods allow users to submit jobs, bind to already submitted jobs, and cancel jobs on remote computers. Other methods allow users to determine whether or not they can submit jobs to a specific resource (through a Globus gatekeeper) and to monitor the job status (pending, active, failed, done, and suspended).

A Grid may comprise more than one GRAM, each of them controlling a set of resources. By means of control or management we defer operations such as submission, monitoring, pausing or stopping. The job manager is created by the gatekeeper located on the remote computer and is responsible for starting and monitoring the job as well as for sending back to the client information regarding the changes in the job's status. A job manager exists for every client request and consists of a common component and a machine-specific component.

- **RSL**

The Resource Specification Language (RSL) [16], which is a structured language for specifying the resource requirements and parameters, is also parsed by GRAM. A gatekeeper is a process running as root on the server before any requests are sent from the client machine and its tasks are:

- Mutual authentication with the client
- Mapping the remote user to a local one
- Activating a job manager on the local host as a local user
- Pass the allocation arguments to the job manager

When the job is finished, the job manager sends the status information back to the client and terminates.

- **GASS**

Global Access to Secondary Storage (GASS) simplifies the porting and running of applications that use file I/O, eliminating the need to manually log onto sites and ftp files or to install a distributed file system. Globus provides an essential subset of GASS services to support the copying of files between computers (servers to client) on which the Grid Services are installed.

2.4 Information Services

The information is used for storing and retrieving information. Directory services are accessible via a network protocol. The Lightweight Directory Access Protocol, LDAP, is a directory service defining an information model and a protocol for querying and manipulating information in the directory. LDAP also include a hierarchical namespaces that defines the organization of the information. LDAP schemas specify what attributes the directory should contain. Each attribute has one type and zero or more values. For each such schema, there exists an information provider that generates the values for each attribute. These information providers are shell scripts. The values provided can either be static, e.g., the number of CPUs, or dynamic, e.g., the number of jobs in the queue of a batch system. Both the protocol and the information model defined in LDAP are extensible.

An information service for a grid environment must be able to handle a wide range of queries and many different types of resources. Furthermore, resource status and dynamic changes in VO membership must be handled as well as dynamic addition and deletion of information sources. LDAP, and other existing directory services such as X.500 [17] and Universal

Description, Discovery and Integration (UDDI) [18], do not fully meet these requirements. The design of Monitoring and Discovering Services (MDS), is an attempt to fulfill these requirements [19].

MDS consists of two parts, the Grid Resource Information Service, (GRIS), and the Grid Index Information Services (GIIS). MDS relies heavily on LDAP, both the GRIS and the GIIS are implemented as OpenLDAP [20] server back ends. Each grid resource runs a GRIS server that advertises static and dynamic information about the resource. Examples are CPU type, current load and available disk space. GIIS server registers themselves to one or more GIIS servers. GIIS servers can either dispatch incoming resource information requests to the appropriate GRIS server or cache information from each GRIS server for faster client access. Resource discovery is done by contacting a GIIS server and retrieving a list of available resources containing contacts to the GRIS server of each resource. All communication between a client and an MDS server is authenticated using the GSI infrastructure.

MDS supports a hierarchical structure for GIIS similar to the Domain Name Servers hierarchy. An example of a hierarchical structure is presented in Figure 2.4 where GRIS (on host B) registers GIIS (on host A) registers GIIS (on host B).

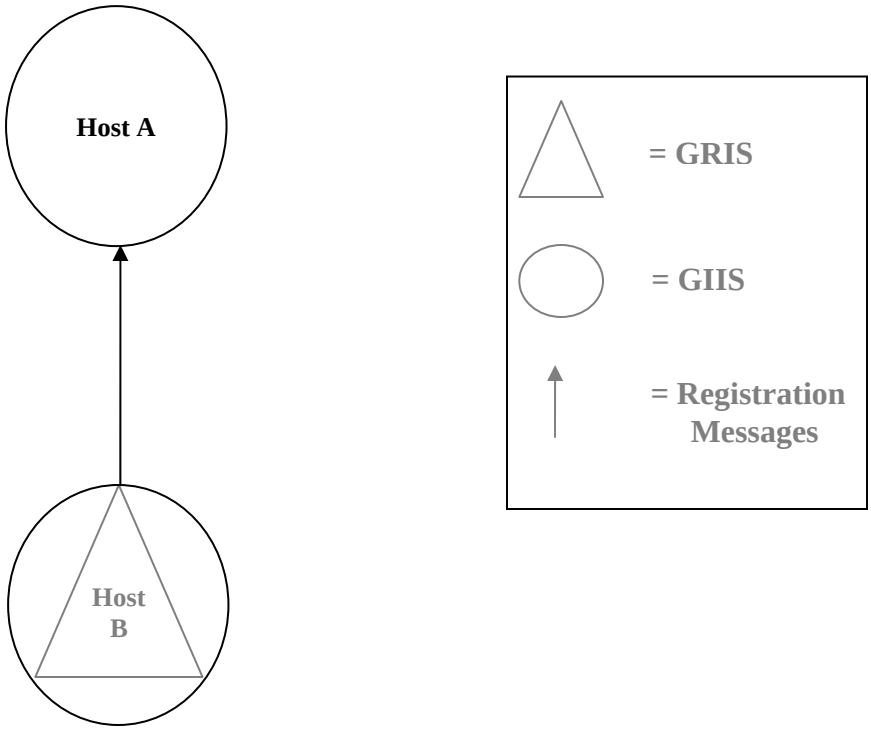

Figure 2.4: Overview of hierarchical GIIS structure

2.5 Data management

The data management [21] pyramid provides support to transfer files among machines in the grid and for the management of these transfers

.

Data transfer: GridFTP [22] is a universal Grid data transfer and access protocol that provides a secure and reliable data transfer among grid nodes. It gives the members the possibility to act as a server or a client. This involves utilities such as GridFTP and globus-url-copy, which are used to move files between grid enabled. GridFTP is based on the FTP protocol [RFC 959] and provides a file transfer service with linked with grid security mechanisms. GridFTP Is the protocol proposed for all data transfers on the Grid. It extends the standard FTP protocol with facilities such as multistreamed transfer; auto tuning and globus based security. GridFTP must support Grid Security Infrastructure (GSI) and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality.

Data access: the Global Access to Secondary Storage (GASS) subsystem provides the access to remote files. It allows programs to use the C standard I/O library to read, write files from remote computers. Copies of remote files opened for reading or writing are maintained in a local file cache with a database that keeps track of the local file name, access mode, URL and reference count.

Data replication: Globus Replica Management (GRM) [23] architecture is responsible for managing complete and partial copies of data sets. Data replication of great scientific interest as valuable data might be copied to several local storage to certify faster access.

Chapter 3

GridFTP: A Data Transfer Protocol for the Grid

In Grid environments, access to distributed data is typically as important as access to distributed computational resources. Distributed scientific and engineering applications require:

- *transfers* of large amounts of data (terabytes or megabytes) between storage systems.
- *access* to large amounts of data (gigabytes or terabytes) by many geographically distributed applications and users for analysis, visualization, etc.

The lack of standard protocols for transfer and access of data in the Grid has led to a fragmented Grid storage community. Users who wish to access different storage systems are forced to use multiple protocols and/or APIs, and it is difficult to efficiently transfer data between these different storage systems.

A Data transfer and access protocol called GridFTP was proposed that provides secure, efficient data movement in Grid environments. The word GridFTP can refer to a protocol, a server, or a set of tools. GridFTP is a fast, efficient, secure, and robust protocol for data transfer. This protocol is in wide use in Grid applications. This protocol, which extends the standard FTP protocol, provides superset of the features offered by the various Grid storage systems currently in use. We choose the FTP protocol because it is the most commonly used protocol for data transfer on the Internet, and of the existing candidates from which to start it comes closest to meeting the Grid's needs.

3.1 Motivation for a common transfer mechanism

There are already a number of storage systems in use by the Grid community. These storage systems have been created in response to specific needs for storing and accessing large datasets. They each focus on a distinct set of requirements and provide distinct services to their clients.

For example, some storage systems (DPSS, HPSS) focus on high-performance access to data and utilize parallel data transfer streams and/or striping across multiple servers to improve performance[24,25]. Other systems (DFS) focus on supporting high volume usage and utilize dataset replication. The SRB system connects heterogeneous data collections and provides a uniform client interface to these repositories, and also provides metadata for use in identifying and locating data within the storage system [26]. Still other systems (HDF5) focus on the structure of the data, and provide client support for accessing structured data from a variety of underlying storage systems [27].

Most of these storage systems utilize incompatible, an often unpublished protocol for accessing data, and therefore require the use of their own client libraries to access data. The use of multiple incompatible protocols and client libraries for accessing storage effectively partitions the datasets available on the grid. Applications that require access to data stored in different storage systems must either choose to only use a subset of storage systems, or must use multiple methods to retrieve data from the various storage systems. It would be mutually advantageous to both storage providers and users to have a common level of interoperability between all of these disparate systems: a common but extensible underlying data transfer protocol. Storage providers would gain a broader user base, because their data would be available to any client. Storage users would gain access to a broader range of storage systems and data. In addition, these benefits can be gained without the performance and complexity problems of the layered client or gateway approach.

Establishing a common data transfer protocol would eliminate the current duplication of effort in developing unique data transfer capabilities for different storage systems. A pooling of effort in the data transfer protocol area would lead to greater reliability, performance, and overall features that would then be available to all distributed storage systems.

3.2 Characteristics of Data Transfer Protocol

Common data transfer protocol should be attractive to users and developers of existing storage systems, which offers a superset of the features offered by systems currently in

regular use. The protocol must be extensible, in order to support future innovations by storage system users and developers.

FTP protocol is the protocol most commonly used for data transfer on the Internet, and the most likely candidate for meeting the Grid's needs. It is attractive in particular for the following reasons.

- It is a widely implemented and well-understood IETF standard protocol. There is a large code base and expertise from which to build.
- It provides a well-defined architecture for protocol extensions, and supports dynamic discovery of the extensions supported by a particular implementation.
- Numerous groups have added various extensions through the IETF. Some of these extensions are particularly useful in the Grid.
- It support transfer between client and server.
- It supports third party transfer between servers.
- The separation of data and control channels onto different sockets allows for easier extensibility for parallel and striped transfers, efficiently transiting firewalls, etc.

Most current FTP implementations support only a subset of the features defined in the FTP protocol (RFC 969) [28] and its accepted extensions. Some of the seldom implemented features are useful to Grid applications. But the standards also lack several features which Grid applications require.

We intend to select a subset of the existing FTP standards and further extend it, adding the following features. We believe that the resulting protocol will be a suitable candidate for the common data transfer protocol for the grid, which we call "GridFTP".

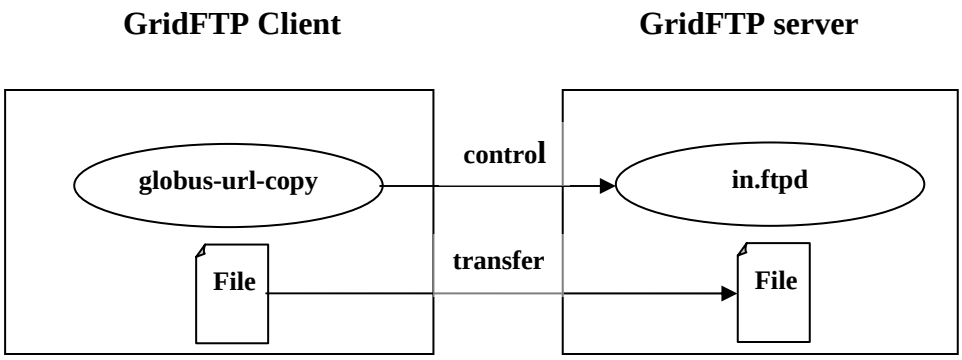

Figure 3.1: Standard FTP Client Server Model

3.3 GridFTP Functionality

The GridFTP provide following types of data transfer [29, 30] in grid computing environment:

- **Grid Security Infrastructure (GSI) and Kerberos support**

Robust and flexible authentication, integrity, and confidentiality features are critical when transferring or accessing files. GridFTP must support GSI and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality. GridFTP provides this capability by implementing the GSSAPI authentication mechanisms defined by RFC 2228, “FTP Security Extensions”.

- **Third-party control of data transfer**

In order to manage large data sets for large distributed communities, it is necessary to provide third-party control of transfers between storage servers. GridFTP provides this capability by adding GSSAPI security to the existing third-party transfer capability defined in the FTP standard.

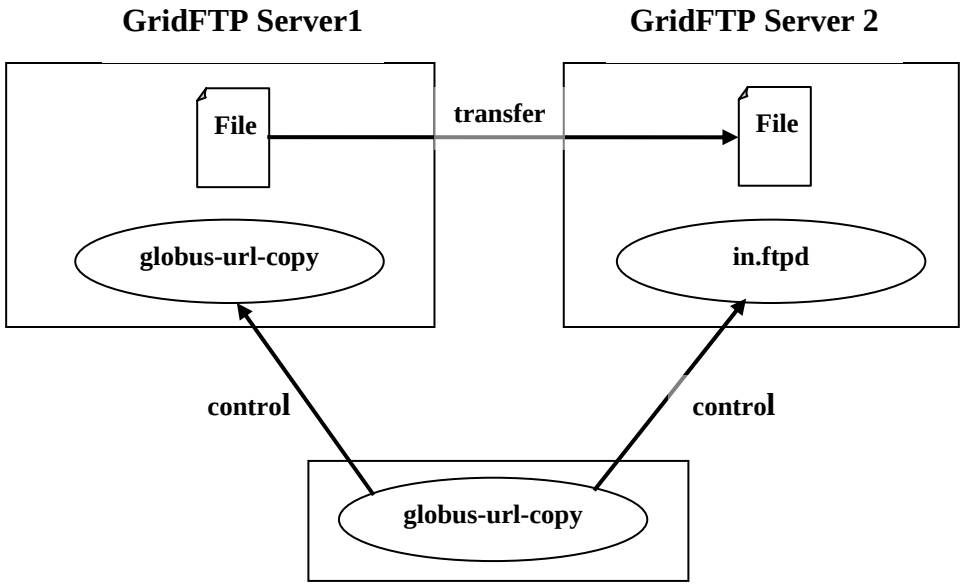

Figure 3.2: Third Party Transfer

- **Parallel data transfer**

On wide-area links, using multiple TCP streams (even between the same source and destination) can improve aggregate bandwidth over using a single TCP stream. This is required both between a single client and a single server, and between two servers. GridFTP supports parallel data transfer through FTP command extensions and data channel extensions defined in the Grid Forum draft.

- **Striped data transfer**

Using multiple TCP streams to transfer data that is partitioned across multiple servers can further improve aggregate bandwidth. GridFTP supports striped data transfer through extensions defined in the Grid Forum draft.

- **Partial file transfer**

Many applications require the transfer of partial files. However, standard FTP requires the application to transfer the entire file, or the remainder of a file starting at a particular offset. GridFTP introduces new FTP commands, as defined in the Grid Forum draft, to support transfers of regions of a file.

- **Automatic negotiation of TCP buffer/window sizes**

Manually setting TCP buffer/window sizes is an error-prone process (particularly for non experts) and is often simply not done. GridFTP extends the standard FTP command set and data channel protocol to support both manual setting and automatic negotiation of TCP buffer sizes both for large files and large sets of small files.

- **Support for reliable data transfer**

Reliable transfer is important for many applications that manage data. Fault recovery methods for handling transient network failures, server outages, etc. are needed. The FTP standard includes basic features for restarting failed transfer that are not widely implemented. The GridFTP protocol exploits these features, and extends them cover the new data channel protocol.

Chapter 4

Establishment of Grid Computing Environment

"Grid" computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation.

This chapter describes the implementation of grid computing environment which include basic requirements for setting up a grid computing environment, how to setup an initial grid and how to maintain and expand the grid.

Before enabling grid environment some of the planning considerations that should be taken into consideration [31]. This includes:

- Planning for installation
- Planning for security
- Planning for related software
- Planning a production environment
- Planning a development environment

4.1 System Requirements

The hardware and software required for the Globus Toolkit 2.4 are described as follows:-

- **Hardware**

In order to build, install, and run the Globus Toolkit on system, the following hardware are taken into consideration:-

CPU

The Globus software itself is not CPU intensive, but the computing power required to run the Globus Toolkit depends on what kind of host system used .

Physical Memory

The Globus Toolkit itself is not memory intensive; therefore, the hosts on which it will run need only have a nominal amount of memory for the sake of the Globus Toolkit code.

Disk Space

Disk space requirements for building, installing, and deploying the Globus Toolkit can vary depending on the number of architectures and the number of development libraries that are built. Thus only approximate disk space requirements can be given.

- **Software**

The software on which the Globus Toolkit depends and what additional software is recommended and why are as follows:

In order to install and run the Globus Toolkit, SSLeay and OpenLDAP are used.

SSLeay. Globus Toolkit can be compiled using SSLeay. The Globus Security Infrastructure (GSI) is implemented in terms of a Generic Security Service Application Program Interface (GSSAPI) that is built on top of the SSLeay package.

OpenLDAP. The Globus Toolkit uses modified libraries from the OpenLDAP distribution, though this will change in the future as the Globus Toolkit modifications are rolled back into OpenLDAP.

4.2 Installation Considerations

There are several issues need to be consider before beginning to install the Globus Toolkit. They are as follows:

4.2.1. Choosing a Host

Choosing the host on which the Globus Toolkit will be installed depends on how it will be used. If no resources will be contributed to the Globus grid resource pool, then the host on which the Globus Toolkit is installed is a matter of convenience. In that case, it is assumed that the Globus Toolkit will be used primarily for something like single sign-on

authentication or to provide access to other resources in the Globus grid. Therefore, no special requirements are needed of the host on which the Globus Toolkit will be installed. If the host on which the Globus Toolkit is installed is to be made available as part of the Globus grid resource pool, then consider issues such as computing power, disk space, and memory. However, running the Globus Toolkit on the same resource is acceptable because the Globus Toolkit is not CPU intensive and should not have an impact on any Globus grid jobs requesting use of this host.

4.2.2 Choosing File systems

There are three considerations for choosing the appropriate file systems for system on which to build, install, and deploy the Globus Toolkit:

1. The Globus Toolkit is better installed on a *shared* file system.
2. Each host running a Globus Toolkit gatekeeper must be able to accommodate deployment of the Globus Toolkit on its local file system.
3. The deployment location must have room for the log files.

4.2.3. Contributing resources

Contribution of resources to the Globus grid resource pool depends on the impact of Globus Toolkit job requests on these resources. There are some methods:

- **Local Site Policies**

Many sites have policies regarding the use of their resources. These policies will have an impact on the choice of resources to contribute and the extent to which they can be made available. Consider the limits on the amount of physical resource such as CPU, memory, and disk; limits on the number of jobs a user can submit or run; and the cost and level of service. Consider how and if these policies can be enforced on Globus Toolkit jobs.

- **Level of Service**

GRAM provides a convenient way of submitting and monitoring jobs remotely. The level of service provides for Globus Toolkit jobs is determined as it would be for any other application running.

- **Accounting**

Resource usage accounting is presumed to be handled locally by each site. The Globus Toolkit does not change any existing local accounting mechanisms. Globus Toolkit jobs run under the user account as specified in the grid-mapfile. Therefore, a user is required to already have a conventional Unix account on the host to which the job is submitted.

4.2.4 Information Services

Globus Toolkit include two specialized types of information services:

GRIS

The GRIS is a distributed information service that can answer queries about a particular resource by directing the query to an information provider deployed as part of the Globus services on a grid resource. Examples of information provided by this service include host identity (e.g., OS and versions), as well as more dynamic information such as CPU availability. Each resource on which the Globus Toolkit is installed will run a GRIS. Therefore, each GRIS is responsible for providing information only about the resource on which it is running. By default, a GRIS is automatically configured and will be assigned to use port 2135.

GIIS

The GIIS represents a centralized MDS server that provides information about all of resources. The normal configuration for an organization would be to have a GRIS on each resource running Globus and one GIIS managed by the lead site for that organization.

4.2.5. Security

The Globus Toolkit uses an authentication system known as gssapi_ssleay, the Generic Security Service API based on Eric A. Young's implementation of Secure Sockets Layer (SSL). This system uses the RSA encryption algorithm[] for its encryption, therefore employing both public and private keys.

4.2.5.1 X.509 Certification Process

The gssapi_ssleay authentication relies on an X.509 certification process. Globus Toolkit users place their X.509 certificates in their home directories, thus identifying themselves to the system.

- **User Certificates and Keys.** The X.509 certificate includes information about the duration of the permissions, the RSA public key, and the signature of the Certificate Authority (CA). The certificates can be created only by the CA, who reviews the X.509 certificate request submitted by the user and accepts or denies it according to an established policy.

- **Gatekeeper Certificates and Keys.** The gatekeeper also must have a certificate and key. They are requested and created in a like manner by the system administrator using the Globus Toolkit certificate request generation script.

- **Proxies**: The gssapi_ssleay authentication requires the use of proxies, a convenient mechanism for reducing the number of times users must enter their pass-phrase. Proxy files must be kept secure, within system's local file system, rather than on the Network File System (NFS). They must allow only the user to have read-access to them, as they essentially allow job submission without pass-phrase protection, a feature that can potentially compromise the security of the system.

  A proxy consists of a new certificate and a private key. The key pair that is used for the proxy, i.e. the public key embedded in the certificate and the private key, may either be regenerated for each proxy or obtained by other means. The new certificate contains the

owner's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the owner, rather than a CA. The certificate also includes a time notation after which the proxy should no longer be accepted by others. Proxies have limited lifetimes.

4.2.5.2 Environment Variables

The Globus Toolkit locates the certificate and key files in the user's ~/.globus directory by default or by referring to environment variables. The basic environment variables that can be set:

GLOBUS_LOCATION

The GSI libaries use GLOBUS_LOCATION as one place to look for the trusted certificates directory. The location $GLOBUS_LOCATION/share/certificates is used if X509_CERT_DIR is not set and /etc/grid-security and $HOME/.globus/certificates do not exist.

GRIDMAP

This environment varibale can be used to override the default location of the grid-mapfile, which is normally /etc/grid-security/grid-mapfile.

X509_CERT_DIR

This environment variable can be used to override the default location of the trusted certificates directory, which is normally /etc/grid-security/certificates.

X509_USER_DELEG_PROXY

This environment is set by the GSI libaries to point at the location of credentials that it recieves during delegation. Application servers usually then copy this value to 509_USER_PROXY and users generally never see it. Setting this value has no effect.

X509_RUN_AS_SERVER

If this environment variable is set(to any value) it causes the GSI libaries not to look for a proxy credential unless X509_USER_PROXY is explicity set. The intent is for this to be used with servers that should always use a given certificate and private key.

X509_USER_CERT

This environment variable can be used to override the default location of the certificate file. For users this is normally $HOME/.globus/usercert.pem. For servers this is normally /etc/grid-security/hostcert.pem.

X509_USERS_KEY

This environment variable can be used to override the default location of the private key file. For users this is normally $HOME/.globus/userkey.pem. For servers this is normally /etc/grid-security/hostkey.pem.

X509_USER_PROXY

This environment variable can be used to override the default location of the user proxy credential, which is /tmp/ *x509up_u<uid>*.

X509_CERT_FILE

File in which one or more trusted certificates are stored (not normally used)

4.3 Required Software

Globus Toolkit Version 2.4 is used for the setup of grid. Globus Toolkit supports Red Hat Linux 9.0 [32] on xseries.

The list of required files to be downloaded [33]:

- Globus Packaging Technology
  gpt-3.0.1-src.tar.gz
- Globus client

globus-all-client-2.4.3-i686-pc-linux-gnu-bin.tar.gz

- Server bundle
  globus-all-server-2.4.3-i686-pc-linux-gnu-bin.tar.gz
- Certificate Authority
  globus_simple_ca_bundle-latest.tar.gz
- Network Time Protocol
  ntp-4.1.1-1.i386.rpm

4.4 Lab environment

This section provides an overview of the configuration of the software and hardware used in our lab. It is a simple Grid environment. We used an Ethernet LAN with four server xSeries machines named alpha, beta, gamma, delta and one client zeta on the LAN. We made alpha server a certificate authority (CA Server). These machines were installed with the Red Hat 9.0 Linux distribution. Figure 4.1 illustrates this environment with the host names and the functionality of each machine.

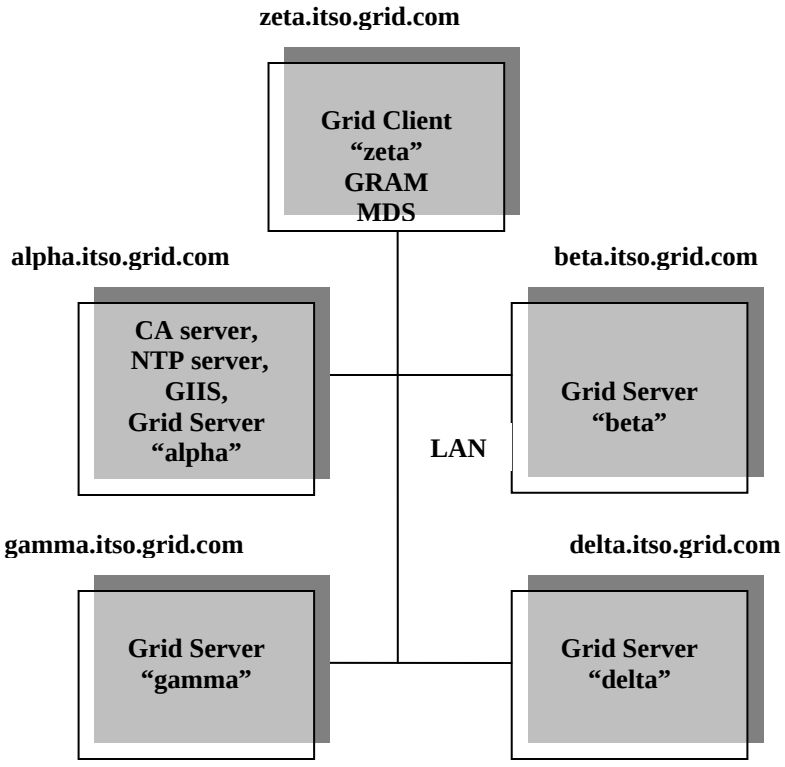

Figure 4.1: Hardware environment and software functions of each machine

4.4.1 Naming and addressing

Table 4.1 summarizes the names of the machines, their IP addresses, the Linux distribution used, and their primary functions.

Table 4.1: Host names and IP addressing

Host name	IP	Linux distribution	Function
alpha.itso.grid.com	192.168.10.201	Red Hat 9.0	CA server, NTP server, Grid Server
beta.itso.grid.com	192.168.10.224	Red Hat 9.0	Grid Server
gamma.itso.grid.com	192.168.10.204	Red Hat 9.0	Grid Server
delta.itso.grid.com	192.168.10.223	Red Hat 9.0	Grid Server
zeta.itso.grid.com	192.168.10.221	Red Hat 9.0	Grid Client

Table 4.2 describes the distinguished name used for the Certificate Authority in our environment:

Table 4.2: CA distinguished name and passphrase

Certificate Authority DN	Passphrase
cn=my test CA, ou=alpha.itso.grid.com, ou =grid	*********

The distinguished name (DN) and passphrase will be used by the Certificate Authority to sign certificate requests.

Table 4.3 describes suggested user and group IDs and passwords.

Table 4.3: User ID and group ID

User ID	Group ID	alpha password	beta password	gamma password	delta password	zeta password
root	root	***	***	***	***	***
globususer	globus	***	***	***	***	***
snobol	snobol	***	***	***	***	***
adminca	adminca	***	No-id	no-id	no-id	no-id

The root ID is used on all machines. A cell containing “no-id” means that the corresponding machine does not have that user ID installed on it. The globususer ID is used to run jobs on the grid for the user and to FTP files. The snobol ID is used to submit jobs to the grid. The adminca ID is used to receive certificate requests for the Certificate. The certificates will be signed using the root ID on machine alpha.

4.5 Setting up the Linux requirements

This section describes the steps that are required to install the Linux environment for using the Globus toolkit 2.4. The major steps to set up this environment are:

4.5.1 Linux Setup

The Linux operating system is used in many ways including support for networking, software development, servers and desktop platforms and is considered as a low cost alternative to other operating systems.

Install Linux on all machines that will be part of the grid. In our lab, we install Red Hat 9.0 on five machines choosing the default installation distributions with no firewall protection so

that network requests (such as RPC) would not be hindered when we needed to access the infrastructure server.

4.5.2 Configure Network Time Protocol (NTP)

For the grid to work properly, the system clocks must be synchronized using NTP server. GSI certificates use GMT and is very sensitive to the time The grid security process creates proxy certificates that are valid for specific times. If the system clocks are not synchronized, the proxy certificates may appear as if they have expired and users may not be able to use the grid. In our lab environment we used an NTP server on machine alpha.

4.6 Installation and Configuration Globus

1. Declare where to install Globus via the environment variable GLOBUS_LOCATION; define GPT_LOCATION to indicate where to install gpt.
2. Install gpt and the necessary client and/or server packages according to the Globus installation instructions.

4.6.1 Setup of Own certificate Authority

The Globus Project provides the Globus Simple CA, convenient way of setting up a certificate authority. Script was installed on alpha to set up a new SimpleCA. Run this script once per Grid.

$GLOBUS_LOCATION/setup/globus/setup-simple-ca

This command creates a new CA 1024-bit RSA private key (CA.key) and certificate (CA.cert) with lifetime 1825 days.

- ***Configure the subject name***

This script prompts information about the CA .The unique subject name for this CA is:
cn=my test CA, ou=alpha.itso.grid.com, ou=demotest, o=Grid

Table 4.4: CA Name components

cn	Represents "common name". Identifies this particular certificate as the CA certificate within the "GlobusTest/simpleCA-hostname" domain, which in this case is my test CA.
ou	Represents "organizational unit". Identifies this CA from other CAs created by SimpleCA by other people. The second "ou" is specific to your hostname (in this cases demotest).
O	Represents "organization". Identifies the Grid.

- ***Configure the CA's email and pass phrase***

The email used by CA, receive certificate requests. It should be real email address of the administrator not the address of users.

It requests a pass phrase to protect the key, The passphrase of the CA certificate will be used only when signing certificates (with grid-cert-sign).

- ***Confirm generated certificate***

A self-signed certificate has been generated for the Certificate Authority with the subject:

/O=Grid/OU=demotest/OU=alpha.itso.grid.com/CN=my test CA

The private key of the CA is stored in :

/home/globus/.globus/simpleCA//private/cakey.pem

The public CA certificate is stored in:

/home/globus/.globus/simpleCA//cacert.pem

The distribution package built for this CA is stored in:

/home/globus/.globus/simpleCA//globus_simple_ca_11116_setup-0.13.tar.gz

The number 1116 is known as CA hash. It will be an 8 hexadecimal digit string.

- ***Complete setup of GSI***

To finish the setup of GSI, run the script on each of machine as root:

$GLOBUS_LOCATION/setup/globus_simple_ca_<hash>_setup/setup-gsi –default

4.6.2 Host Certificate

In order to use resources in a grid, host must first request and install security certificates from a reputable certificate authority (CA).

Request and sign a host certificate and then copy it into the appropriate directory for secure services. The certificate must be for a machine which has a consistent name in DNS; you should not run it on a computer using DHCP where a different name could be assigned to computer.

- ***Request a host Certificate***

On each of the server machines (alpha, beta , gamma, delta) as root , the administrator issue the command.

grid-cert-request -host <hostname of requesting machines>

This creates the following files:

- /etc/grid-security/hostkey.pem
- /etc/grid-security/hostcert_request.pem
- /etc/grid-security/hostcert.pem(empty)

- ***Signing host Certificate***

CA machine (alpha) sign the request certificate as root, the administrator issue the command.

grid-ca-sign -in /root/hostcert_request.pem -out /root/hostcert.pem

The file hostcert.pem contains the certificate and should be sent back to the user and should be saved in the directory /etc/grid-security/certificates. The administrator should verify the identity of the user. The certificate should be owned by the user, and read-only for other users.

4.6.3 User Certificates

For each user using the grid, must request user certificates which will sign using the *globus* user.

- ***Request a user Certificate***

User snobol on machine zeta, run the command:

grid-cert-request

This creates the following files:

- /home/snobol/.globus/userkey.pem
- /home/snobol/.globus/usercert_request.pem
- /home/snobol/.globus/usercert.pem(empty)

- ***Signing user Certificate***

CA machine (alpha) sign the request certificate as root, the administrator issue the command.

grid-ca-sign -in /root/usercert_request.pem -out /root/usercert.pem

The file usercert.pem contains the certificate and should be sent back to the user and should be saved in the directory /etc/grid-security/certificates. The administrator should verify the identity of the user. The certificate should be owned by the user, and read-only for other users.

4.6.4 Adding a Grid mapfile entry

Certificates have been signed and installed, users must be added to the grid mapfile so that they can access resources on a grid host. The mapping consists of associating a grid user's DN with a local user on the host.

4.6.5 Testing

To verify the Simple CA is installed in /etc/grid-security/certificates and that certificate is in place with the correct permissions, run:

user$ grid-proxy-init -debug –verify

A Full Proxy has been created by grid-proxy-init which can be use to perform various grid operations.

4.7 Setting up the gatekeeper

On each server (alpha, beta , gamma, delta) , aappend to /etc/services the following lines.

```
gsigatekeeper   2119/tcp        # Globus Gatekeeper
gsiftp          2811/tcp        # GsiFTP
```

In the file /etc/xinetd.d/gsigatekeeper on each server, containing the lines:

```
service gsigatekeeper
{
 socket_type =  stream
 protocol     =   tcp
 wait          =   no
 user          =   root
 env           =   LD_LIBRARY_PATH=${GLOBUS_LOCATION}/lib
 server        =   ${GLOBUS_LOCATION}/sbin/globus-gatekeeper
 server_args  =    -conf \ ${GLOBUS_LOCATION}/etc/globus-gatekeeper.conf
 disable       =    no
}
```

In the file /etc/xinetd.d/gsiftp on each sever, containing the lines:

```
service gsiftp
{
 instances          = 1000
 socket_type        = stream
 wait               = no
 user               = root
 env                = LD_LIBRARY_PATH=${GLOBUS_LOCATION}/lib
 server             = ${GLOBUS_LOCATION}/sbin/in.ftpd
 server_args        = -l -a -G ${GLOBUS_LOCATION}
 log_on_success  += DURATION USERID
 log_on_failure   += USERID
 nice               = 10
 disable            = no
}
```

4.8 Setting up MDS

Monitoring and Discovery Service (MDS) is based on OpenLDAP, allowing to create own configuration of hierarchical GIIS. MDS have one Grid Information Index Service (GIIS) in the alpha machine, which collects the data reported by the Grid Resource Information Servers (GRIS) in all of the machines. The GRIS servers send information about their respective servers to the GIIS. The user will be able to query the GIIS from the zeta client machine.

To set up this structure, modification of several configuration files is be done. These files name the GIIS and GRIS, and show how these components should register with each other. Figure 9-2 shows the relationship among the MDS components:

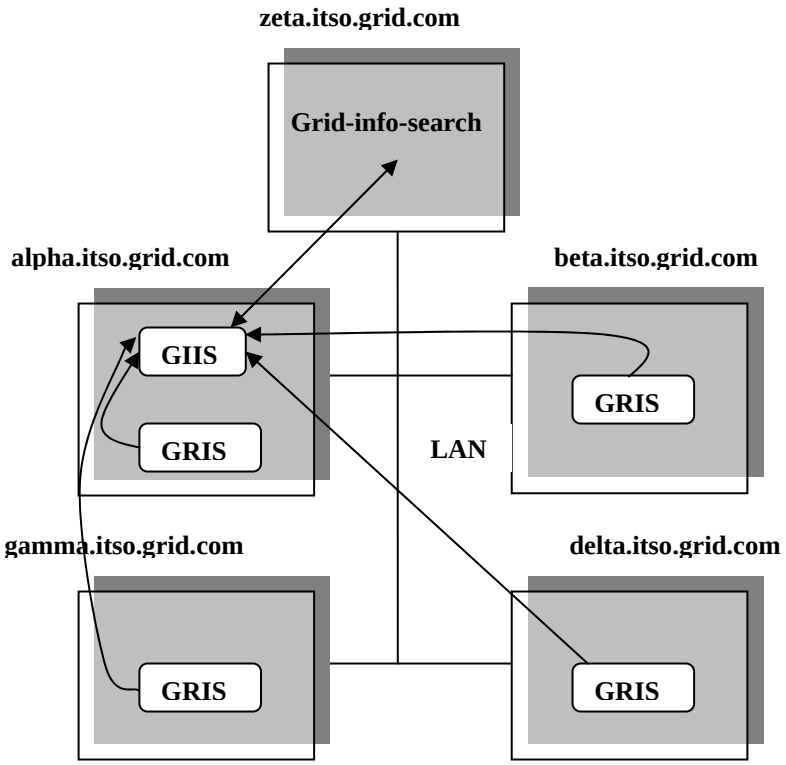

Figure 4.2: MDS Configuration

GIIS are set on the alpha machine and GRIS are set on all the other server (alpha, beta , gamma , delta).

4.8.1 MDS on client

Modiified the $GLOBUS_LOCATION/etc/grid-info.conf file lines as shown below so that the searches go to the GIIS on machine alpha.

GRID_INFO_HOST = “alpha.itso.grid.com”

GRID_INFO_ORGANIZATION_DN”=”Mds-Vo-name=alpha.itso.grid.com, o=Grid”

4.8.2 Secure MDS

This MDS permits anonymous access. The grid-info-search command should use the -x flag to indicate an anonymous search request. The MDS can be secured so that only certified users can access the GIIS and only certified server GRISs can register to send information to the GIIS. Each resource will have its own LDAP information service that can be connected to remotely for obtaining system and status information. The GIIS can only be configured to run on a host on which the Globus Toolkit will be installed.

4.8.2.1 Request a LDAP certificate

On each of the server machines (alpha, beta , gamma, delta) as root , the administrator issue the command.

grid-cert-request –service ldap -host <hostname of requesting machines>

This creates the following files:

- /etc/grid-security/ldap/ldapkey.pem
- /etc/grid-security/ldap/ldapcert_request.pem
- /etc/grid-security/ldap/ldapcert.pem(empty)

4.8.2.2 Signing LDAP certificate

CA machine (alpha) sign the request certificate as root, the administrator issue the command.

grid-ca-sign -in /root/ldapcert_request.pem -out /root/ldapcert.pem

The file ldapcert.pem contains the certificate and should be sent back to the user and should be saved in the directory /etc/grid-security/ldap. The administrator should verify the identity of the user. The certificate should be owned by the user, and read-only for other users.

4.9 Verification

4.9.1 Server interface

Installation on each machine can be check as root using the command:

$GPT_LOCATION/sbin/gpt-verify

GRAM can be check by listening on their port:

netstat –an | grep 2119

The command used to check the secure MDS are:

globus-mds start

globus-mds stop

4.9.2 Client interface

On client machine (zeta) , logged on as the user snobol and sets up the enviroment so that globus command can be issued by the user.This line is be added to one's login profile:

. $GLOBUS_LOCATION/etc/globus-user-env.sh

Proxy is created with the command:

grid-proxy-init

Client interface for GRAM , MDS and Grid FTP are discussed below

- **GRAM**

GRAM has the following client commands to submit and manage jobs on the grid environment.

- **globus-job-run**

This is an online interface for job submissions. It is the easiest command to use to submit a job and returns the output of its result.

The basic command syntax is:

globus-job-run <contact string> <command>

where <contact string> specifies a machine's host name, port, and service to which to send the request.

The syntax of a contact string is host:port/jobmanager-name. The default port is 2119 and the default job manager's name is "jobmanager".

- **globus-job-submit**

This is an interface for batch job submissions. It will immediately return with an URL (with the job contact string embedded) that can be use to query the status of job. The command is similar to *globus-job-run*, but the *globus-job-submit* command does not return the output of its result. To obtain the output, run the job management commands *globus-job-status*, *globus-job-get-output*, and *globus-job-clean/globus-job-canc1e* pointing to the URL generated as result of the *globus-job-submit* execution.

The basic command syntax is as follows:
globus-job-submit <contact string> command

- **globusrun**

This is a command that gives access to the RSL, the language which provides a common interchange to describe resources [31]. The *globus-job-run* and *globus-job-submit* commands are both shell script wrappers around *globusrun.*

The basic command syntax is:
globusrun <contact string> <RSL>

- **globus-job-status**

This is a job management command that returns a job status of one of the following:

- pending
- active
- done
- failed
- others

- **globus-job-get-output**

This is a job management command that collects the output when the job finishes.

- **globus-job-clean/globus-job-cancel**

This is a job management command that stops the job if it is running, and cleans up the cached copy of the output.

➢ **MDS (GRIS and GIIS)**

MDS has a client command to query for details about resources in the grid environment.

- **grid-info-search**

This command sends one or more queries to GRIS and GIIS and displays the result in the standard output. The queries are RFC1558 compliant with the LDAP search filter, since the command embeds the *ldapsearch* command.

The basic command syntax is:

grid-info-search [options] <filter> [attributes...]

➢ **GridFTP**

GridFTP provides a client command to copy files between local and remote locations.

- **globus-url-copy**

The basic command syntax is:

globus-url-copy [options] <source URL> <destination URL>

Where:

<source URL> is the URL to source file, or '-' for standard input.

<destination URL> is the URL of the destination file, or '-' for standard output.

4.10 Setting up grid applications

In grid platforms, setting up a grid application should be very straight forward and should not require additional remarks about non-trivial issues. Setting up grid applications that the grid administrator should be aware of are as follows:

4.10.1 Deploying an application

Grid applications are single instruction multiple data (SIMD) programs. Most of the computing demanding applications have this feature and that, in a loosely coupled distributed system, the data-parallelism tends to be more efficiently exploited. The deployment of an application has two distinct phases:

Code deployment This phase is performed when the application is first deployed to the grid or when the code is modified and has to be updated.

Data deployment This phase has to be performed every time a new execution is issued. Data deployments are more time-consuming, more frequent, and while they are being performed, the application stands idle waiting for the data to arrive. For this reason, there are a few things that are worth mentioning when discussing application deployments:

- Some grid platforms make it possible for multiple applications to be executed simultaneously; if this is the case, application deployments do not cause much impact, as the grid does not have to be idle while they are performed.

- Some few applications are capable of dealing with streaming data, and some grid platforms do support this sort of application (the application start processing the data as soon as it gets to the nodes). If a single-application grid is to be set up and its application works this way, adopting a streaming enabled grid platform is something to consider.

- Deployments should be ultimately performed by the system administrator, but the platform might make facilities available for the application developers to submit their application code and data so that every deployment is correctly logged and assigned to its developer.

Application deployments should not be a serious concern in terms of performance. For a well-behaved grid application, the processing time has to be much greater than the communication time.

4.10.2 Making application data available

Deploying the application data may be performed in several ways. If the application relies on centralized data-base servers, there must be a platform tool or even an application task for fetching the data at the server, partitioning it conveniently, and sending the pieces to the grid nodes. This automated process is usually the best option when the grid application is integrated with legacy systems that store their state on data bases, but other issues arise when deciding how to spread the application data across the grid.

- **Web publishing**

Probably the simplest way to make data available to grid applications is to publish it on the ordinary Web sites of FTP servers. There is a whole generation of systems and tools to aid developers to accomplish this task efficiently, but this philosophy has some major drawbacks. If publishing the data itself is easy, getting it to process may not be; the grid application programmer will have to deal with network programming to build its application, which is not desired; additionally, depending on how the application is designed, it can suffers from scalability, as every node may try to access the data at once. This happens because the responsibility for distributing the data across the grid relies on the application designer, and not on the grid platform.

- **Data-base server oriented**

It differs from the previous scenario in the sense that many legacy systems already have their data stored on data-base servers. The main drawbacks also apply to this case: The application developers will probably have to deal with database access issues when developing their applications.

- **Grid platform driven**

When the grid platform provides facilities for fetching the application data and distributing it across the platform automatically, the application development process can be drastically simplified. In this case, the platform must include tools for describing the data in its original source and specifying how it should be partitioned and distributed among the grid nodes. Scalability issues remain totally under control of the platform itself.

This is certainly the best option among all, but so far there isn't any grid platform that provides full-fledged facilities for fetching and distributing application data across its nodes.

Chapter 5
Design and implementation

GridFTP is a fast, efficient, secure, and robust protocol for transfer of large data files across the grid and is an evolution of the FTP protocol [6]. This protocol is wide use in Grid applications, primarily through an implementation available in the Globus Toolkit. The Globus Toolkit provides a server, a script-capable command-line interface client called *globus-url-copy*, and a set of development libraries. While *globus-url-copy* will meet the needs of most data movement tasks, at times custom code is the only solution. Different application can be enable to access remote files stored behind a GridFTP server.

Security extensions to FTP were first proposed in [7]. GridFTP adopts these extensions and adds further enhancements such as:

- Transfer efficiency eg. parallelism and striping.
- Reliability eg. Automatic restarting after a network failure.
- Third party transfer eg. initiating from site ‘A’ a file transfer between sites ‘B’ and ‘C’.

FTP and GridFTP both operate under a client/server model, where the server runs at the ‘remote’ site. With FTP, two TCP connections are established between the client and server - one for control messages, and another for the actual data. GridFTP introduces the possibility of multiple data connections, which is how it can achieve greater efficiency. While GridFTP is used primarily as a transport layer between higher-level applications, its additional commands (such as the ability to list files and make directories) also allow GridFTP servers to be treated as secure data repositories.

5.1 Design consideration

5.1.1 Software

GridFTP is technically the name of the protocol, this is also used as the software which implements the protocol. The software operates under a client/server model as drawn in Figure 1, but in many cases a site may be both a client and a server.

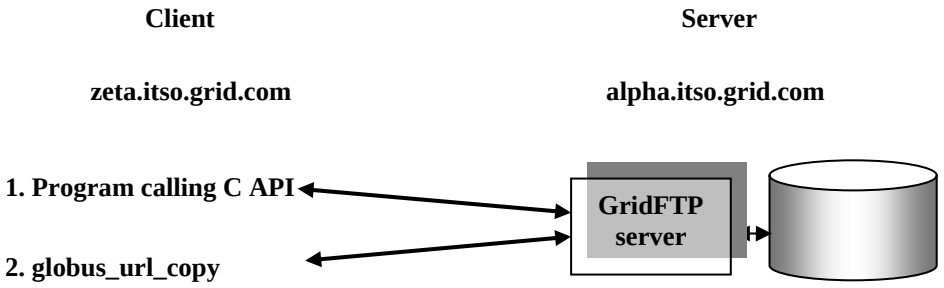

Figure 5.1: Client/Server model in GridFTP

5.1.1.1 Server

Any site may set itself up as a GridFTP data repository by running the GT2.4 GridFTP server. If it is not already running as a result of starting Globus, the server may be started manually using the following procedure [9]:

1. Obtain a valid proxy using grid-proxy-init
2. Then start the GridFTP server :

% $GLOBUS_LOCATION/sbin/in.ftpd -s -p 5678

The -s tells in.ftpd to run as a daemon.

The -p specifies the port number to listen on.

5.1.1.2 Client

At the client site there is a C API and a higher-level command-line tool called globus-url-copy which is a utility for transferring known files between a client and a server (or between two servers). The prefix of the source file name should be gsiftp:// for GridFTP transfers.

5.1.2 Interface

The main interface to GridFTP is the client side C API [34], from which higher-level applications such as globus-url-copy can be built.

The Globus Toolkit provides two primary libraries for GridFTP: the control library and the client library. The control library provides very low-level primitives for command processing, parallel stream (multiple TCP streams) I/O, and so forth. This library gives extreme flexibility, but it requires a deep understanding of the GridFTP protocol.

The Globus FTP Client library provides a convenient way of accessing files on remote FTP servers. In addition to supporting the basic FTP protocol, the FTP Client library supports several security and performance extensions to make FTP more suitable for Grid applications. The client library is built on top of the control library and hides this complexity. Each of the GridFTP operations represents a complete session. The protocol and state machine are completely hidden. In addition to protocol support for grid applications, the FTP Client library provides plugin architecture for installing application or grid-specific fault recovery and performance tuning algorithms within the library. Application writers may then target their code toward the FTP Client library, and by simply enabling the appropriate plugins, easily tune their application to run it on a different grid.

This module is0 for building applications. It includes functions for the following:

File Manipulation

Check the existence of a file or directory on the server	globus_ftp_client_exists(…)
Make a directory on the server	globus_ftp_client_mkdir (…)
Remove a directory from the server	globus_ftp_client_rmdir(…)
List the files on a server	globus_ftp_client_list(…)
Move (rename) a file on the server	globus_ftp_client_move(…)

Data Transfer

Pull the contents of a file from the server	globus_ftp_client_get(…)
Pull part of the contents of a file fro the server	globus_ftp_client_partial_get(…)
Push data to a file on the server	globus_ftp_client_put(…)
Push data to part of a file on the server	globus_ftp_client_partial_put(…)
3rd party transfer between two servers	globus_ftp_client_third_party_transfer()

5.1.3 Date Set Identification

The C API functions take a URL string as an argument to identify the target file (or directory) on the server. For example, to specify the file /home/snobol//abc.doc on server alpha.itso.grid.com, the URL is:

gsiftp://alpha.itso.grid.com/home/snobol/abc.doc

5.2 Implementation

This section describes the implementation of an application with the Grid FTP C API.

5.2.1 Environment

GLOBUS_LOCATION environment variable has been set to the root directory of the machine's Globus installation.

Run the command globus-user-env.sh to set any other Globus variables.

5.2.2 Makefile Header

A header for the Makefile can be generated automatically using the globus-makefile header tool. The resulting header file can be included by the Makefile to set variables of libraries and paths etc that the FTP client module requires. The following script prepares a header called 'globus_header'.

#! /bin/bash

Create a makefile header that captures all the dependencies needed by the globus_ftp_client module

Syntax: globus-makefile-header globus_ftp_client -flavor=gcc32dbg -link=static > globus_header

5.2.3 Makefile

A Makefile can be based on the one below. The C code is in ‘pullfile.c’ and the resulting executable is called ‘pullfile’. It is the ‘globus_header’ file which defines the GLOBUS_PKG_LIBS variable (among others).

A *static* link is being performed here for executable portability. This requires some extra work in the Makefile. Firstly, when linking statically, explicit mention must be made of GLOBUS_LIBS which includes various standard libraries such as 'socket', 'nsl'.

Secondly, even though the GLOBUS_PKG_LIBS are being pulled in statically, the GLOBUS_LIBS must be pulled in dynamically as some of them (eg. 'dl') have no static equivalent.

```
# Makefile for GridFTP program.
include globus_header
CC=gcc
CFLAGS=-c -x c++ -g -O2 -Wall $(GLOBUS_INCLUDES)
LD=gcc
LDFLAGS=$(GLOBUS_LDFLAGS) pullfile.o -Wl,-Bstatic $(GLOBUS_PKG_LIBS) -Wl,-Bdynamic $(GLOBUS_LIBS)
all: pullfile
pullfile: pullfile.o
$(LD) $(LDFLAGS) -o pullfile
pullfile.o: pullfile.c
$(CC) $(CFLAGS) -o pullfile.o pullfile.c
```

clean:

rm pullfile *.o

5.3 Programming Environment

5.3.1 Parallel Transfer

Parallel transfer is where a number of simultaneous TCP connections are used to transfer the data between the client and the server, as illustrated in Figure 5.2.

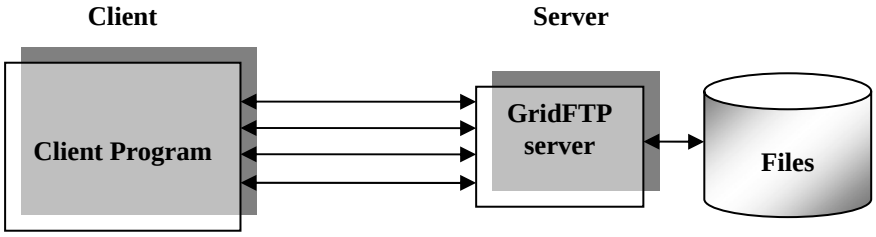

Figure 5.2: Parallel TCP data connection

This can potentially result in a faster transfer because:

- If a TCP connection is held up in the network (eg. due to dropped packets), the other connections may still be able to proceed.
- The different connections could be routed differently, resulting in higher overall bandwidth.

The degree of parallelism can be set with the following function:

globus_ftp_client_operationattr_set_parallelism(…)

Parallel transfer is only supported in extended transfer mode.

5.3.2 Striped Transfer

It is possible to set up a GridFTP server as a ‘striped’ server. This means that files on the server are split into pieces (stripes) and stored separately, perhaps on different discs. This can

improve the overall speed at which data can be retrieved from disc and transmitted by the server.

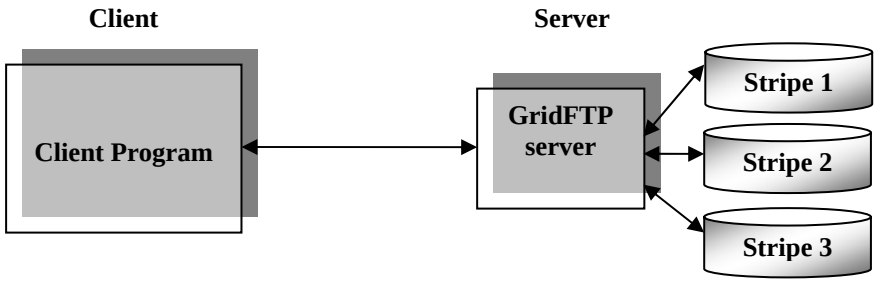

Figure 5.3: A striped server can store pieces of a file on separate discs

This is transparent to the client. However, when pushing a file to the server, the client can specify how the file should be split into stripes. This is known as specifying the striping 'layout' for the file. It can be set with the following function:

globus_ftp_client_operationattr_set_layout(…)

5.3.3 Third-Party Transfer

The GridFTP API allows a client to initiate a file transfer between two servers, as shown in Figure 5.4.

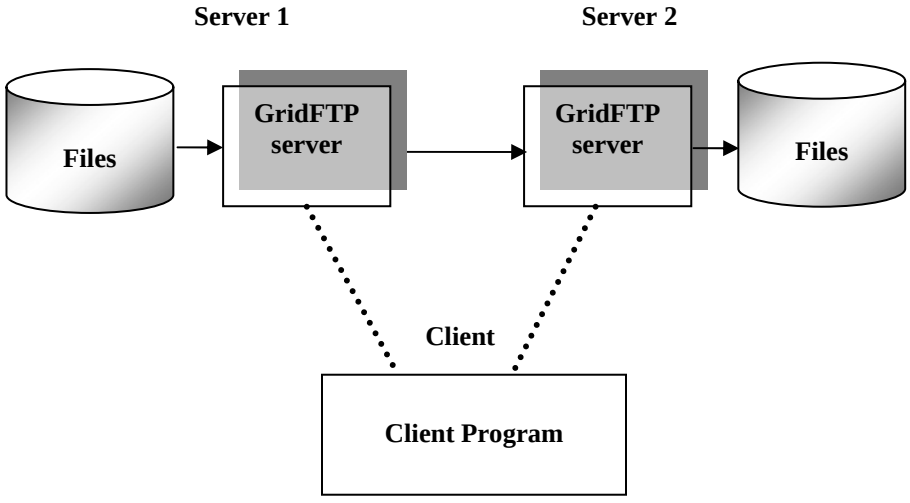

Figure 5.4: Third party transfer between two GridFTP servers

Presumably the client user must have a gridmap file entry at both Server 1 and Server 2 for the transfer to be successful. The transfer is initiated using the function below:

globus_ftp_client_third_party_transfer(…)

The two key arguments to this function are the URLs of the source file at Server 1 and the destination file or directory at Server 2.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Grid computing is a powerful paradigm that can solve the technology needs of developing environments by providing access to industry standard software and hardware technologies at a fraction of the cost. Grid computing provides the necessary environment to share applications, data, network resources, storage and processing capacities through strict policies and guaranteed quality of service models. Grid infrastructure has evolved to a point where several commercial as well as non-commercial versions of the grid are now available for academic institutions to evaluate and deploy immediately.

This Dissertation covered the central parts of Grid architecture, and Globus toolkit, a widely used implementation of Grid middle-ware. The aim of this Dissertation was to familiarize with components of a Grid Architecture and to exhibit the benefits every single user could make out of it. A particular focus of the Globus effort is the development of a small grid based on toolkit providing essential services that can be used to implement a variety of higher-level programming models, tools, and applications. As we have explained in this brief review, Globus components have been deployed in large scale and used to implement a variety of applications. Grid Computing can now be seen as a field located at the meeting point of many highly technical and delicate fields such as Security, Distributed Systems and Programming.

We have described a new open source implementation of the GridFTP protocol. In designing this system, we set out to create a robust, performant, and modular data transfer framework for use in a variety of data-intensive tools and applications. The resulting Globus GridFTP system integrates a variety of techniques, including a modular protocol processing pipeline and parallel I/O, to meet its design goals in way that no other system has done before.

6.2 Future work

Today, grid systems are still at the early stages of providing a reliable, well performing, and automatically recoverable virtual data sharing and storage. We will see products that take on this task in a grid setting, federating data of all kinds, and achieving better performance, integration with scheduling, reliability, and capacity.
Here are some other interesting issues regarding the future of grid computing:

- The grid expansion may embrace multiple media types; thus, radios, televisions, and phone networks will also be available as a grid service.
- Personal and home-based offices will become a reality; this may change the way that small and large corporations are conceived.

These are some of the possibilities that might arise from the grid world, and there is no doubt that they will definitely change the way that we deal with information in our personal and professional activities.

There are also additional features and developing technologies that would be useful, such as , pipelining of commands, web services(SOAP) , etc.

GridFTP does not have internal support for accounting of transfer details. An additional accounting related infrastructure would be required.

The GridFTP server does not schedule data transfer as such. It processes requests as they arrive. So some scheduling technique can be proposed.

References

[1] Andrew S. Tanenbaum , Maarten van Steen Distributed Systems: Principles and Paradigms, 3/e, 2004.

[2] Foster and C. Kesselman, editors. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco, California, 1999.

[3] C. Kesselman I. Foster and S. Tuecke. The Anatomy of the Grid: Enabling ScalableVirtual Organizations. *International Journal Supercomputer Applications,* 15(3), 2001.

[4] Ian Foster. What is the grid? A three point checklist. Grid Today, 1(6), July 2002.

[5] Rajkumar Buyya, Muthucumaru Maheswaran, et al. A taxonomy and survey of grid resource management systems for distributed computing. The Journal of Software Practice and Experience, v.32 n.2, pp.135–164, 2002

[6] The Globus Project: A Status Report. I. Foster, C. Kesselman. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998.

[7] C. Kesselman. I. Foster. Globus: A metacomputing infrastructure toolkit. In *International J. Supercomputer Applications*, page 115. 1997.

[8] http://www.globus.org/. The globus alliance.

[9] I. Foster, C'kesselman, and S.Trecke. A secure architecture for computational grids. In Proc. 5th ACM Conference on Computer and Communications Security Conference, pages 83-82, 1998.

[10] Security for Grid Services. V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke. *T1welfth International Symposium on High Performance Distributed Computing (HPDC-12)*, IEEE Press, to appear June 2003

[11] J. Hirsch. Introducing SSL and certificates using SSLeay. World Wide Web Journal, 2(3 Summer) , 1997.

[12] Douglas R. Stinson. Cryptography, theory and practice. CRC press, New York, 1995.

[13] X.509 Proxy Certificates for Dynamic Delegation. V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, F. Siebenlist. *3rd Annual PKI R&D Workshop*, 2004.

[14] Grid Resource Management. J. Nabrzyski, J.M. Schopf, J. Weglarz (Eds). Kluwer Publishing, Fall 2003.

[15] Czajkowski K., et al, *Resource Management Architecture for Metacomputing Systems*, in The 4th workshop on Job Strategies for Parallel Processing. 1998. pp 62-82.

[16] The Globus resource specification language rsl v 1.0. Internet, May 2004. http://www-fp.globus.org/gram/rsl_specl/html.

[17] R. Butler Von Welch, D. Engbert, I. Foster, S. Tuecke, J. Volmer, and C. kesselman. A National Authentication Infrastructure. IEEE Computer Society Press, 2000.

[18] Inc. The Stencil Group. The evolution of UDDI. Internet, May 2004. http://www.stencilgroup.com/ideas_scope_200207uddiv3.pdf

[19] K. Czajkowski, S.Fitzgerald, I. Foster, and C.Kesselman. Grid information services for distributed resource sharing. In Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10). IEEE CS Press, Aug. 2001.

[20] OpenLDAP. http://www.openldap.org. Internet, May 2004

[21] Data Management and Transfer in High Performance Computational Grid Environments. B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. Parallel Computing Journal, Vol 28 (5), May 2002, pp. 749-771.

[22] The Globus alliance . Grid FTP universal data transfer for the grid.Internet, May 2004. http://www-fp.globus.org/datagrid/deliverable/C2WPdraft3,pdf

[23] Performance and Scalability of a Replica Location Service. A.L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, R. Schwartzkopf. *Proceedings of the International IEEE Symposium on High Performance Distributed Computing (HPDC-13)*, June 2004.

[24] BTierney, W. Johnston, J. Lee, G. Hoo, and M. Thompson. End-to-end performance analysis of high speed distributed storage systems in wide area ATM networks. In *NASA/Goddard Conference on Mass Storage Systems and Technologies*, 1996, LBNL-39064.

[25] 1R.W. Watson and R.A. Coyne. The parallel I/O architecture of the high- performance storage system (HPSS). In *IEEE MSS Symposium*, 1995.

[26] Information on SRB is available on the World Wide Web at http://www.npaci.edu/DICE/SRB

[27] Information about HDF/HDF5 is available on the World Wide Web at http://hdf.ncsa.uiuc.edu/.

[28] FTP (File Transfer Protocol), http://www.w3.org/Protocols/rfc969/Overview.html

[29] GridFTP Protocol Specification (Global Grid Forum Recommendation GFD.20). W. Allcock, editor. March 2003.

[30] GridFTP Update January 2002**.** W. Allcock, J. Bresnahan, I. Foster, L. Liming, J. Link, P. Plaszczac. *Technical Report, January 2002.*

[31] The Role of Planning in Grid Computing. J. Blythe, E. Deelman, Y. Gil, C. Kesselman, A. Agarwal, G. Mehta, K. Vahi. *ICAPS 2003,* 2003.

[32] Red Hat Linux 9.0 specification www.redhat.com

[33] Globus Toolkit 2.4 Install www.globus.org/download/gt2.4/

[34] Globus Toolkit API Documentation • www.globus.org/developer/api-reference.html

[35] Ahmar Abbas Grid Computing: A practical guide to technology & Application.

[36] A book on Grid Computing by Jushy Joseph Craig Fellenstein.

[37] A book on Grid Computing: " Making the Globus Infrastructure a reality " by Fran Berman , Geoffrey C. Fox , Anthony. J. G. Hey.

Appendix A (Commands)

This appendix has descriptions of the commands available in the Globus Toolkit. The commands are grouped together by key areas of Globus Toolkit functionality.

Security Commands

Command	Description
grid-cert-request	Creates a new certificate request and private key.
grid-cert-info	Displays certificate information. Unless the optional -file argument is given, the default location of the file containing the certificate is assumed to be the $X509_USER_CERT. If X509_USER_CERT not set $HOME/ .globus/ usercert.pem
grid-cert-renew	Creates a new key and renewal request for a Globus certificate. The Globus Certificate Authority (CA) will notify a user when the user's certificate is about to expire. The notification message also contains a challenge (a text string) that grid-cert-renew will embed in the renewal request, for higher security.
grid-change-pass-phrase	Changes the "pass phrase" that protects your private key. If the -file argument is not given, the default location of the file containing the private key is assumed to be $X509_USER_KEY If X509_USER_KEY not set, $HOME/.globus/userkey.pem
grid-check-ca-policyfile	grid-check-ca-sig <signer> <subject> Obtains policy file (EACL) from the default location, parses it, builds the GAA internal policy structure and checks if the specified signer is allowed to sign certificates for specified subject.
grid-inquire-policyinfo	grid-inquire-policy-info <signer>. obtains policy file (EACL) from the default location, parses it, builds the GAA internal policy structure, finds ACL entry corresponding to the signer and returns a list of authorized rights and corresponding conditions, if any.
grid-proxy-init	Creates a proxy certificate that can be used for authentication without having to enter the protecting pass-phrase.

Command	Description
grid-proxy-info	Displays proxy certificate information. Common name of the subject (/C=US/O=Globus/ CN=user1) ; common name of the proxy generator; proxy information: (limited of full proxy); strength of proxy (number of bits used in the key); and remaining time. It can also work in another mode, verifying that it is a valid proxy, which means that the following requirements are met: the proxy exists, the proxy is not a limited proxy, the proxy has not expired and is active for another H hours, and the proxy is encrypted using at least B bits Unless the optional -file argument is given, the default location of the file containing the proxy certificate is assumed to be the $X509_USER_PROXY. If X509_USER_PROXY not set, tmp/x509up_uXXX, where XXX is the local UID#.
grid-proxy-destroy	Removes any user proxy certificates generated using grid-proxy-init. It can also be used as a general-purpose "safe remove" program, in that it opens the file and replaces the information contained therein with garbage before deleting it. If no arguments are given, the proxy file at the default location (or at the location pointed to by env .var X509_USER_PROXY, if set) will be destroyed.
grid-security-config	This script will ask some questions about site specific information. This information is used to correctly generate the Grid Security Infrastructure for your site.

Job Submission Commands

Command	Description
globusrun	Run a single executable on a remote site. The job startup is done using the GRAM or UROC Globus services. Also, the GASS service can be used to provide access to remote files and or redirecting standard output streams. In addition to starting jobs, `globusrun` can be used to list previously started jobs or do authentication tests to GRAM gatekeepers.
globus-setup-test	Verifies credentials setup. Also verifies whether the user has submission capabilities to a certain gatekeeper. If no arguments are given, all gatekeepers on the local host will be tested.
globus-job-cancel	globus-job-cancel Cancels a job previously started using globus-job-submit.
globus-job-run	Allows the user to run a job at one or several remote resources. It translates the program arguments to a RSL request and uses globusrun to submit the job.
globus-job-clean	Kills the job if it is still running and cleans the information concerning the job.
globus-job-get-output	For the job specified, gets the standard output or standard error resulting of the job execution.
globus-job-status	Display the status of the job. See also globus get-output to check the standard output or standard error of your job
globus-job-submit	For batch job submission (i.e., submitting a job to a queue via some local scheduling manager). Allows the user to submit a job to a remote resources, using the same 1command-line syntax as `globus-job-run` does. The program translates the program arguments into an RSL request and uses `globusrun` to submit the job in batch submission mode, that is, after the job is submitted, no connection exists between the local host and the remote host. `globus-job-submit` prints out a job id after successful submission of the job to a remote resource. Unless stdout/stderr are specified, the program output will be buffered at the remote site and can be retrieved at any time with `globus-job-get-output`. If the output is buffered, the user must make sure to run `globus-job-clean` when the program output is no longer needed.

Information Services Commands

Command	Description
grid-info-add	Modifies the GIS server based on the contents of input file. adds one or several objects to the MDS, according to the LDIF entries in the file pointed by the -file argument. If the -file argument is omitted, the program assumes this information is available on stdin. To be able to store information in the MDS, the password of the Directory Manager for the local organization must be provided. If the password argument is not given, the password will be asked for interactively by grid-info-add.
grid-info-create	Modifies the MDS server based on the contents of the input file.
grid-info-genfilter- template	To provide a bootstrapping of scripts that all follow the same convention. And to remove the burden from the developer by inserting the appropriate code to perform input validation and to guarantee output conformance.
grid-info-host	Creates a set of CLDIF entries for the host '${bindir}/globus-hostname' where options are: -usage (Displays this message); -version (Displays the current version number); -f[ile] hostfile (Creates only CLDIF entries for host listed in hostfile); -all (Creates CLDIF entries for all hosts at the site)…
grid-info-hostsearch	Queries a GRIS on a specified host and port. By default the local host GRIS and associated default GRIS port of 2135 are used.
grid-info-hostsearch	Queries a GRIS on a specified host and port. By default the local host GRIS and associated default GRIS port of 2135 are used.
grid-info-interfaces	For the current host, build the associated GlobusNetworkInterface and GlobusNetwork Interface Image templates.
grid-info-modify	Update an existing database entry. Metadata is inserted by grid-info-update.

Command	Description
grid-info-networks	For the current site, build the associated GlobusNetwork and GlobusNetworkImage templates. For the current site, build the associated network objects based on the built computational resource nodes for the site.
grid-info-prep	To convert CLDIF to LDIF with simple format and error checking. This script is directly tied to the format and structure of a "commented LDIF" template file.
grid-info-remove	See grid-info-add. deletes one or more objects to the MDS, according to the LDIF entries in the file pointed by the -file argument. If the -file argument is omitted, the program assumes this information is available on stdin. To be able to delete information from the MDS, the password of the Directory Manager for the local organization must be provided. If the -password argument is not given, the password will be asked for interactively by `grid-info-remove`.
grid-info-search	Searches the GIIS. Sends one or more queries to the GIIS and displays the result on stdout. The query QUERY is a RFC 1558 compliant LDAP search filter. By default, the object's name and all its attributes are displayed: this can be narrowed down by specifying what attributes to display after the query.
grid-info-site	Usage: $PROGRAM_NAME [hostname | -f[ile] hostfile]. Creates a set of CLDIF entries for the host \"hostname\, where options are: -usage (displays this message); -version (displays the current version number); -f[ile] hostfile (creates only CLDIF entries for host listed in hostfile).The hostfile contains a list (one per line) of host names and an optional location of the Globus bin directory. The default for this directory is $bindir, e.g., host-dns-name [bindir]

Commands	Descriptions
grid-info-update	See grid-info-add. modifies one or more objects to the MDS, according to the LDIF entries in the file pointed by the -file argument. If the -file argument is omitted, the program assumes this information is available on stdin. To be able to delete information from the MDS, the password of the Directory Manager for the local organization must be provided. If the -password argument is not given, the password will be asked for interactively by `grid-info-remove`.
grid-mapfile-addentry	can be used to add entries to the mapfile. For example the following command would add a user to the mapfile: gridmapfile- add-entry -dn "/C=US/O=Globus/O=State University/CN=Joe User" –ln juser. You must type in the distinguished name exactly as it appears in the certificate. Not doing so will result in an authentication failure
grid-mapfilecheck-consistency	checks the consistency of the Grid mapfile. Options:-help, -usage (displays help); -version (displays version); -mapfile FILE, -f FILE Path of gridmap to be used.
grid-mapfiledelete-entry	deletes an entry from the Grid mapfile. Options: -help, -usage (displays help); -version (displays version); -dn <DN> Distinguished Name (DN) to add; -ln <local name> Local Login Name (LN) to map DN to; -dryrun, -d (shows what would be done but will not delete the entry; -mapfile file, -f file (path of gridmap file to be used)

Other Tool Commands

Commands	Descriptions
config-guess	Program provided under GNU license from Free Software foundation that attempts to guess a canonical system name.
globus-gass-server	A utility that allows the user to start up a stand-alone GASS server, to which he can upload or download files from locally accessible file systems. When no "Enable" or "Disable" options are specified, the globus-gass-server runs with the default options of -l , -t -u, -r, -w, -o, and -e. By default, the globus_gass_server will output the base URL it is listening on and then remain in the foreground.
globus-gass-servershutdown	This command allows the user to shut down a previously started (remote) GASS server, *but* the GASS server must have been started with the -c command line option (client-destroy) in order for this command to take effect.
globus-hostname	This is a simple shell script that acts like the Unix hostname. Returns the system hostname and make some additional checks to ensure a fully qualified hostname. Setting the environment variable GLOBUS_HOSTNAME to a non-null string will cause globushostname to return that value instead. This is useful for specifying the use of certain network interfaces when communicating etc.
globus-hostname2contacts	Converts a hostname to a list of resource manager contact strings. Returns contact strings on stdout, identifying services that provided by gatekeeper(s) running on the host specifed. If no type or service is specified, the program will return matching GRAM jobmanager services, in the following order: contacts with job manager types other than fork and poe; contacts with job manager type poe; and contacts with job manager type fork. A contact string is on the form: "host:port/servicename:certificate subject".

Commands	Descriptions
globus-gass-cachedestroy	Will destroy (clean without coherence check) the Globus cache on all he machines listed in ~/.globus/my-contacts, or in <file> if -f option used, or only on the machine specified with the t option.
globus-list-my-contacts	Creates a list of machine (gatekeeper contacts) on which the user has "globus access".
globus-netstat	Hides the implementation-specifics of netstat and reformats the output to be consistent across architectures, producing a subset of Unix System V netstat output.
globus-sh-exec	Sources the globus-sh-tools file, then executes a user script. Allows the user to run a script adhering to the Bourne shell format on a remote machine without having to worry about correct paths to various Unix commands. To facilitate this, globus-sh-exec provides to the user script a set of GLOBUS_SH_PROG variables, pointing to the location of program PROG. About 125 commands are currently defined in this way. In addition the to the GLOBUS_SH_* variables, the variable bindir is defined, pointing to the Globus tools bin directory. The variables sbindir, libexecdir and a couple more are defined in the same fashion. A rudimentary PATH variable set to cover most system commands is also defined before the script is run. globus-sh-exec also recognizes a GASS URL as script argument, and handles it correctly. Additional optional arguments are passed on to the user provided script.
globus-version	Shows version number. Returns version information of currently installed Toolkit, such as patch level, release date, etc. If no arguments are given, the assumed option is "-string".
globus-deploy-path	globus-deploy-path prints the full path to the local deploy directory, which contains the machine-specific setups etc. The location of the deploy tree is determined as follows, in the following order: the path given by env.var. GLOBUS_DEPLOY_PATH; the path given to globus-local-deploy (only applies to deployed copies of globus-deploypath); and the deploy path given by the GlobusService object in the MDS. If the local deploy directory cannot be located, "<not found>" is returned.

Commands	Descriptions
globus-development-path	Prints the full path to the "development" directory (include files and libraries) that corresponds best to the flavor indicated by the commandline options (e.g., pthreads, debug, and 64-bit) The development subtree in the Globus install directory contains several different "flavors", mostly affecting the communications library Nexus. (Nexus has the ability to use underlying vendor provided high performance communications libraries.) The program auto-senses the architecture on which it is running. If the desired flavor has not been built when installing Globus, "<not found>" is returned. If no arguments are given, "-debug -standard -nothreads -32 -64" is assumed.
globus-install-path	Prints the full path to the Globus install tree. The location of the install tree is as follows, in the following order: the path given by env.var. GLOBUS_INSTALL_PATH and the path that was given as "prefix" when installing Globus. If an installed tree cannot be located, "<not found>" is returned.
globus-personalgatekeeper	Options: -help, -usage (displays usage), -version (displays version), -ebug (displays extra output), -start [-jmtype <type>] [-port <port>] (starts a new gatekeeper, mapping default service to a jobmanager. By default, the jobmanager is configured with jmtype=fork. The option - port can be used to restrict the gatekeeper to use a particular port. The default is to let the system choose a port. -list (scans for active personal gatekeepers. If found, an authentication test is made to determine if the gatekeeper is still functioning. If the authentictation test succeeds, the contact to that gatekeeper is printed out.) directory <contact> (eturns the temporary directory used by the personal gatekeeper associated with <contact>. The directory contains grid-mapfile, log file, etc.; -kill <contact> (Finds and kills the personal gatekeeper associated with <contact>); -killall (finds all personal gatekeepers running on the local host and kills them.)

Commands	Descriptions
globus-rcp	Remote copies using GASS and Globus submission. Many options. Allows the user to copy files to and from remote locations. It attempts to as much as possible mimic the functionality of cp and rcp.
globus-tools-path .	Prints the full path to the tools directory in the Globus install tree, tailored for the current architecture. The program auto-senses the architecture on which it is running. If the tools directory cannot be located, an empty string is returned. The default search for a services directory can be overridden by setting the environment variable GLOBUS_TOOLS_PATH
globus-services-path	Prints the full path to the services directory in the Globus install tree, tailored for the current architecture. The program auto-senses the architecture on which it is running. If the services directory cannot be located, an empty string is returned. The default search for a services directory can be overridden by setting the environment variable GLOBUS_SERVICES_PATH.
globus-tilde-expand	Expands the leading tilde sign (~) (and the specified user name if provided) to the full path of the user's home directory (or of the specified user), the same way the C shell proceeds. This command is intended to be used in command interpreters that do not perform such expansions, such as /bin/sh. The expanded string is returned without any carriage return termination character, to allow easy concatenation in a shell script.
globus-url-copy	Remote file copy using URL syntax Copies a file specified by sourceURL to a location specified by destURL, using the GASS transfer API. All protocols supported by GASS (local file, http, https, ...) are supported. Piping to/from stdin/stdout (setting source/dest argument = '-') is supported.

Appendix B (Source Code)

++

Simple Remote file transfer

++

```
#include <stdio.h>
#include "globus_ftp_client.h"
static globus_mutex_t lock;
static globus_cond_t cond;
static globus_bool_t done;
#define MAX_BUFFER_SIZE 2048
#define ERROR -1
#define SUCCESS 0
static
void
done_cb(
void * user_arg,
globus_ftp_client_handle_t * handle,
globus_object_t * err)
{
char * tmpstr;
if(err)
{
fprintf(stderr, "%s", globus_object_printable_to_string(err));
}
globus_mutex_lock(&lock);
done = GLOBUS_TRUE;
globus_cond_signal(&cond);
globus_mutex_unlock(&lock);
return;
}
static
void
data_cb(
void * user_arg,
globus_ftp_client_handle_t * handle,
globus_object_t * err,
globus_byte_t * buffer,
globus_size_t length,
globus_off_t offset,
globus_bool_t eof)
```

```
{
if(err)
{
fprintf(stderr, "%s", globus_object_printable_to_string(err));
}
else
{
if(!eof)
{
FILE *fd = (FILE *) user_arg;
int rc;
rc = fread(buffer, 1, MAX_BUFFER_SIZE, fd);
if (ferror(fd) != SUCCESS)
{
printf("Read error in function data_cb; errno = %d\n", errno);
return;
}
globus_ftp_client_register_write(
handle,
buffer,
rc,
offset + length,
feof(fd) != SUCCESS,
data_cb,
(void *) fd);
} /* if(!eof) */
} /* else */
return;
}
 /* data_cb */

/**************************
* Main Program
**************************/

int main(int argc, char **argv)
{
globus_ftp_client_handle_t handle;
globus_byte_t buffer[MAX_BUFFER_SIZE];
globus_size_t buffer_length = MAX_BUFFER_SIZE;
globus_result_t result;
char * src;
```

```
char * dst;
FILE * fd;

/*************************************
* Process the command line arguments
*************************************/

if (argc != 3)
{
printf("Usage: put local_file DST_URL\n");
return(ERROR);
}
else
{
src = argv[1];
dst = argv[2];
}

/**********************************
* Open the local source file
**********************************/

fd = fopen(src,"r");
if(fd == NULL)
{
printf("Error opening local file: %s\n",src);
return(ERROR);
globus_module_activate(GLOBUS_FTP_CLIENT_MODULE);
globus_mutex_init(&lock, GLOBUS_NULL);
globus_cond_init(&cond, GLOBUS_NULL);
globus_ftp_client_handle_init(&handle, GLOBUS_NULL);
done = GLOBUS_FALSE;
result = globus_ftp_client_put(&handle,
dst,
GLOBUS_NULL,
GLOBUS_NULL,
done_cb,
0);
if(result != GLOBUS_SUCCESS)
{
globus_object_t * err;
err = globus_error_get(result);
```

```
fprintf(stderr, "%s", globus_object_printable_to_string(err));
done = GLOBUS_TRUE;
}
else
{
int rc;
rc = fread(buffer, 1, MAX_BUFFER_SIZE, fd);
globus_ftp_client_register_write(
&handle,
buffer,
rc,
0,
feof(fd) != SUCCESS,
data_cb,
(void *) fd);
}
globus_mutex_lock(&lock);
while(!done)
{
globus_cond_wait(&cond, &lock);
}
globus_mutex_unlock(&lock);
globus_ftp_client_handle_destroy(&handle);
globus_module_deactivate_all();
return 0;}
```

/++

```
* Operation attributes are used to set a parallelism. 4
* This means the transfer must run in extended block mode MODE E.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++/

#include <stdio.h>
#include "globus_ftp_client.h"
static globus_mutex_t lock;
static globus_cond_t cond;
static globus_bool_t done;
int global_offset = 0;
#define MAX_BUFFER_SIZE (64*1024)
#define ERROR -1
#define SUCCESS 0
#define PARALLELISM 4

/**********************************************************************
* done_cb: A pointer to this function is passed to the call to
* globus_ftp_client_put (and all the other high level transfer
* operations). It is called when the transfer is completely
* finished, i.e. both the data channel and control channel exchange.
* Here it simply sets a global variable (done) to true so the main
* program will exit the while loop.
**********************************************************************/

static
void
done_cb(
void * user_arg,
globus_ftp_client_handle_t * handle,
globus_object_t * err)
{
char * tmpstr;
if(err)
{
fprintf(stderr, "%s", globus_object_printable_to_string(err));
}
globus_mutex_lock(&lock);
done = GLOBUS_TRUE;
globus_cond_signal(&cond);
globus_mutex_unlock(&lock);
return;
}
```

```
/*****************************************************************************
* data_cb: A pointer to this function is passed to the call to
* globus_ftp_client_register_write. It is called when the user supplied
* buffer has been successfully transferred to the kernel.
*****************************************************************************/
static
void
data_cb(
void * user_arg,
globus_ftp_client_handle_t * handle,
globus_object_t * err,
globus_byte_t * buffer,
globus_size_t length,
globus_off_t offset,
globus_bool_t eof)
{
if(err)
{
fprintf(stderr, "%s", globus_object_printable_to_string(err));
}
else
{
if(!eof)
{
FILE *fd = (FILE *) user_arg;
int rc;
rc = fread(buffer, 1, MAX_BUFFER_SIZE, fd);
if (ferror(fd) != SUCCESS)
{
printf("Read error in function data_cb; errno = %d\n", errno);
return;
}
globus_ftp_client_register_write(
handle,
buffer,
rc,
global_offset,
feof(fd) != SUCCESS,
data_cb,
(void *) fd);
global_offset += rc;
```

```
} /* if(!eof) */
else
{
globus_libc_free(buffer);
}
} /* else */
return;
} /* data_cb */

/**************************
* Main Program
**************************/

int main(int argc, char **argv)
{
globus_ftp_client_handle_t handle;
globus_ftp_client_operationattr_t attr;
globus_ftp_client_handleattr_t handle_attr;
globus_byte_t * buffer;
globus_result_t result;
char * src;
char * dst;
FILE * fd;
```
Chapter 7. Using Globus Toolkit for data management **205**
```
globus_ftp_control_parallelism_t parallelism;
globus_ftp_control_layout_t layout;
int i;

/*************************************
* Process the command line arguments
*************************************/

if (argc != 3)
{
printf("Usage: ext-put local_file DST_URL\n");
return(ERROR);
}
else
{
src = argv[1];
dst = argv[2];
}
```

```
/**********************************
* Open the local source file
**********************************/

fd = fopen(src,"r");
if(fd == NULL)
{
printf("Error opening local file: %s\n",src);
return(ERROR);
}

/*************************************************************************
* Initialize the module, handleattr, operationattr, and client handle
* This has to be done EVERY time
* The mutex and cond are theoretically optional,
* NOTE: It is possible for each of the initialization calls below to
* fail and we should be checking for errors.
*************************************************************************/

globus_module_activate(GLOBUS_FTP_CLIENT_MODULE);
globus_mutex_init(&lock, GLOBUS_NULL);
globus_cond_init(&cond, GLOBUS_NULL);
globus_ftp_client_handleattr_init(&handle_attr);
globus_ftp_client_operationattr_init(&attr);

/*************************************************************************
* Set any desired attributes, in this case we are using parallel streams
*************************************************************************/

parallelism.mode = GLOBUS_FTP_CONTROL_PARALLELISM_FIXED;
parallelism.fixed.size = PARALLELISM;
layout.mode = GLOBUS_FTP_CONTROL_STRIPING_BLOCKED_ROUND_ROBIN;
layout.round_robin.block_size = 64*1024;
globus_ftp_client_operationattr_set_mode(
&attr,
GLOBUS_FTP_CONTROL_MODE_EXTENDED_BLOCK);
globus_ftp_client_operationattr_set_parallelism(&attr,
&parallelism);
globus_ftp_client_operationattr_set_layout(&attr,
&layout);
globus_ftp_client_handle_init(&handle, &handle_attr);
```

```
/*****************************************************************
* globus_ftp_client_put starts the protocol exchange on the control
* channel. Note that this does NOT start moving data over the data
* channel
*****************************************************************/

done = GLOBUS_FALSE;
result = globus_ftp_client_put(&handle,
dst,
&attr,
GLOBUS_NULL,
done_cb,
0);
if(result != GLOBUS_SUCCESS)
{
globus_object_t * err;
err = globus_error_get(result);
fprintf(stderr, "%s", globus_object_printable_to_string(err));
done = GLOBUS_TRUE;
}
else
{
int rc;
/*****************************************************************

* This is where the data movement over the data channel is initiated.
* You read a buffer, and call register_write. This is an asynch
* call which returns immediately. When it is finished writing
* the buffer, it calls the data callback (defined above) which
* reads another buffer and calls register_write again.
* The data callback will also indicate when we have hit eof
* Note that eof on the data channel does not mean the control
* channel protocol exchange is complete. This is indicated by
* the done callback being called.
*****************************************************************/


for (i = 0; i< 2 * PARALLELISM && feof(fd) == SUCCESS; i++)
{
buffer = malloc(MAX_BUFFER_SIZE);
rc = fread(buffer, 1, MAX_BUFFER_SIZE, fd);
```

```
globus_ftp_client_register_write(
&handle,
buffer,
rc,
global_offset,
feof(fd) != SUCCESS,
data_cb,
(void *) fd);
global_offset += rc;
}
}

/*****************************************************************
* The following is a standard thread construct. The while loop is
* required because pthreads may wake up arbitrarily. In non-threaded
* code, cond_wait becomes globus_poll and it sits in a loop using
* CPU to wait for the callback. In a threaded build, cond_wait would
* put the thread to sleep
*****************************************************************/

globus_mutex_lock(&lock);
while(!done)
{
globus_cond_wait(&cond, &lock);
}
globus_mutex_unlock(&lock);

/*****************************************************************
* Since done has been set to true, the done callback has been called.
* The transfer is now completely finished (both control channel and
* data channel). Now, Clean up and go home
*****************************************************************/

globus_ftp_client_handle_destroy(&handle);
globus_module_deactivate_all();
return 0;
}
```