ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

GRID COMPUTING – TESTBED AND APPLICATION

BY

ABYOT ASALEFEW

MARCH, 2006

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

GRID COMPUTING – TESTBED AND APPLICATION

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment for the Degree of Master of Science in Computer Engineering

By
Abyot Asalefew

Advisor
Ato Dawit Mengistu

Addis Ababa

March, 2006

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

GRID COMPUTING – TESTBED AND APPLICATION

BY

ABYOT ASALEFEW

Approval by Board of Examiners

Dr. Getachew Birru	_____
Chairman, Department of Electrical and Computer Engineering	Signature
Ato Dawit Mengistu	_____
Advisor	Signature
_____	_____
Examiner	Signature
_____	_____
External Examiner	Signature

CONTENTS

List of Figures

List of Tables

ACKNOWLEDGMENT

First and foremost I offer my thanks to GOD, who made this work possible.

I express my deepest gratitude to my advisors Ato Dawit Mengistu and Dr. Kumudha Raimond for their advice, and suggestions that greatly enriched this thesis.

I have to express my sincere appreciation to the staff of Electrical and Computer Engineering for their encouragement and concern. My special thanks goes to Ato Fetahi Zebenigus for his suggestions which helped me in finalizing this work.

Finally, my heartfelt appreciation and indebtedness goes to my family for their love, patience, support and encouragement throughout my study.

ABSTRACT

Grid computing, which is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources, is one of the currently emerging computing technologies gaining much attention from both researchers and end-users. This thesis is concerned with the study of concepts and components that are used in the deployment of computational Grid, including the testing of its middleware software Globus Toolkit.

The study of Grid concepts and components performed by this thesis work includes the construction of sample Grid testbed, followed by testing its performance by using a sample computationally intensive LM Hash cracking application with customizations for Grid environment.

As the technology becomes more pervasive and our technological world becomes more complex, without proper and timely acquaintance and understanding of the technology, stakeholders of AAU will find themselves at an increasing disadvantage. The work of introducing Grid computing, by deploying sample Grid testbed, to the users, students and researchers of AAU is one of the timely and economically appropriate works in this regard; and hence the other objective of this thesis work.

This thesis also contains a detailed deployment and planning study of Grid construction in the AAU environment. This is further supported by a well-documented step-by-step installation and configuration procedure that is followed in the construction of the sample Grid testbed.

1. INTRODUCTION

1.1 Background

Based on the efforts of individuals and collaborative researchers and theorists, today's science is much more dependent on computation, data analysis and collaboration. The dependency is mainly driven by problem's nature of complexity, bulkiness and urgency. These natures of problems are leading us to a new computing paradigm called Grid computing, mainly because the advent of increasingly powerful technologies, and exponential increase of computer power, data storage and communication, does not live up to the demand of researchers for computational resources.

In Grid computing, the intention is to build an infrastructure that will make distributed computational resources available as easily as electric power is through the electricity distribution grid. Apart from the need of utilizing idle distributed resources, the other motivation for Grid computing came from the problems in processing scientific data, where the use of dedicated supercomputers is expensive and mostly infeasible. Large networks of much cheaper and less powerful processors have long been touted as a natural alternative to such dedicated devices, but there has never been a technology capable of exploiting such distributed computational resources. The aim of Grid computing is to provide such technologies and therefore the challenges in Grid computing lie in developing the software to drive the Grid.

With the intentions of making Grid a reality, the Globus Alliance, an international collaboration, which conducts research and development to create fundamental Grid technologies, is formed. The Globus Alliance is a community of organizations and individuals developing fundamental technologies behind the "Grid," which lets people share computing power, databases, instruments, and other on-line tools securely across corporate, institutional, and geographic boundaries without sacrificing local autonomy. In addition to working on standards and protocols for Grid technology, the alliance team has developed the Globus Toolkit (GTK); which is an open source software toolkit that can be used for building Grid systems and applications.

Software is a vital part of Grid projects. The open source community has produced a wide variety of Grid software components which includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. All these are packaged as a set of components that can be used either independently or together to develop applications. Understanding the capabilities of each component, the strengths and weaknesses of each, and the types of problems that are currently covered (or not covered) by these components can be a bewildering challenge for new Grid projects. And because the GTK has grown through an open-source strategy, similar to that of the Linux operating system, it has encouragement for broader, more rapid adoption and of course a path to greater technical innovation.

The focus of this thesis work is to test and analyze the GTK by deploying it on a few machines of AAU. Setting up Grid testbed followed by testing the functionalities and performances of GTK components is the core and practical part of this thesis work. Testing the GTK by using a sample compute intensive application will help a lot for the very young and new field of research, mainly in the areas of standardization and rectification of software packages and protocols.

Introducing Grid technology to the AAU, so that researchers, students and users of AAU can benefit a lot by effectively utilizing the computing resources in a coordinated and collaborated manner, is also another objective behind this thesis work.

1.2 Problem Description

As can be seen from the trends of computing systems in the real world, people are developing a number of computational hardware resources - compute servers, handhelds, file servers, networks, playstations, cell phones etc… In the mean time scientists, researchers, engineers and users are coming up with compute intensive tasks in order to solve them using the current infrastructure in a space-time constraint, which seems to be very difficult and impractical.

In order to address the above problem a number of approaches like Distributed Computing, Parallel Computing, and Clustering have been devised and implemented. But needs for the

computing resource availability, sharing, efficiency and virtuality are still increasing and demanding for a new scheme of computing; and hence Grid computing came into picture.

By understanding the concepts and working principles of Grid computing, one can contribute to the maturity of the technology by implementing, testing, and utilizing it. While testing and analyzing Grid softwares from Globus Alliance are the main objectives of this work, the other objective is introducing Grid computing to the researchers, students and users of AAU by deploying and testing a sample testbed which can be used as a role model for large scale campus wide deployment.

1.3 Related Work

Currently, a number of researches are being conducted in the areas of Grid computing. Projects like Globus [7] and Legion [19] aim to provide a comprehensive support for Grid computing, where as other projects, like Condor [4], have more focused goals like targeting high-throughput applications which can be used on top of Globus.

Taking only the Globus research wing, a number of GTK versions are developed in a short period of time. By the time of this paper work (period for analysis, software pre-requisite collection, installation, configuration, and testing), GTK3 of Globus was the latest one with the alpha version of GTK4 to be released soon. As can be seen from the history of Globus software new versions with additional features and packages are being released frequently. This is mainly due to the fact that the Globus package is to encompass a wide range of components including security, information repository, data management and communication which are very broad by themselves and are difficult to consider all at once and hence the need for subsequent revisions and new releases.

Because GTK contains a number of large components, these components need a thorough understanding, analysis, deployment and testing. And according to the researcher of this paper the current workings, in this regard, are not satisfactorily enough; and hence the need for further analysis and testing. Further analysis, deployment, and testing of GTK on the AAU computing infrastructure, which had not experienced Grid computing before, and

then testing it by using a sample computationally intensive application, are the contributions of this thesis work.

The computationally intensive test application identified, RainbowCrack LM Hash cracking, has only enjoyed the concept of Philippe Oechslin's "faster time-memory trade-off technique"[21]; but not the parallely available massive computing resources. In the implementation and testing part of this thesis, the current execution flow of RainbowCrack is carefully studied, planned and modified (with parallel execution in mind) so that it can benefit a lot from the availability of massive computing resources from the Grid environment. This customization of RainbowCrack is a new achievement which will help to increase the efficiency and speedup factors in the field of hash cracking.

1.4 Methodology and Thesis Outline

It is the workings of the following methodologies that make up the body of this paper.

1. Literature survey

   This is assessing any published and unpublished works, journals and books to gather any information relevant to the research work. The survey is to be conducted in the libraries and mainly on the internet.

2. Software deployment

   Here the open source Grid software called GTK is to be used in order to setup the Grid testbed over the AAU computing infrastructure.

3. Testing

   Once the testbed is deployed, it is going to be tested, for the functionalities of its components as well as performance gains, by using a sample computationally intensive application.

In this thesis work, it has been tried to explain what Grid computing is all about. Issues as well as tasks and challenges in setting up and utilization of computational Grid are also part of this paper. Though the work is an initial attempt and no previous work has been done in the area of Grid computing in order to deploy Grid softwares and then set up high performance computing lab in AAU, it has been tried to address some of the most common encounters and problems that are faced in deployment of Grid testbed on the few machines of AAU, Department of Electrical and Computer Engineering.

The report has started with an introductory chapter, which highlights the whole thesis work: - problem description, objective and approach.

Following the introductory chapter is the theoretical discussion of Grid computing:- concepts and components. In this chapter, the basic concepts of computing, trends in computing, user's requirements and the responses from the technology are discussed in brief.

In the third chapter of this thesis, special emphasis is given to the Grid software. In this chapter, a detailed discussion of the general design architecture for Grid software is presented. Most of the basic and main components which any Grid software is supposed to contain are discussed. The final section of this chapter is a discussion of specific Grid software GTK; in this section, the different modules of this toolkit and their features are discussed.

Following the conceptual discussion of Grid computing and its middleware software is the practical aspect of this thesis. In this "TESTBED DESIGN AND DEPLOYMENT" chapter, design methodologies and procedures followed for the Grid testbed design are discussed briefly. With this discussion, requirements for deploying GTK and design model for testbed implementation are presented. Detailed discussions and step by step procedures followed in the testbed deployment are presented in Appendix A of this paper.

Utilization of the deployed Grid testbed by using a computationally intensive application, its customization for Grid and its performance enhancement are the discussions of chapter

five. This chapter has also outlined some attributes of jobs or applications that can be considered as deciding factors in selecting an application for Grid environment.

Finally chapter six presents some conclusions and recommendations obtained from this thesis work. With the recommendation portion of this chapter, the researcher of this thesis has highlighted some areas of Grid and/or GTK which need further work, reengineering and cleanup.

In the end, appendices and bibliography are attached. With the appendices, detailed documentation of Grid testbed deployment as well as guideline for using basic Globus commands is presented. The Appendix also contains a preliminary work for deploying large scale production Grid for AAU. In this section, Grid planning for AAU such as requirement analysis, existing computing infrastructure analysis, and possible Grid topology for AAU are discussed in a detailed manner.

2. GRID COMPUTING

2.1 Trends in Computing

Computer systems are undergoing a revolution. From 1945, when the modern computing era began, until about 1985, computers were large and expensive. Even minicomputers cost tens of thousands of dollars each. As a result, most organizations had only a handful of computers, and for lack of way to connect them, these operated independently from one another.

Starting in the mid-1980's, however, two advances in technology began to change that situation. The first was the development of powerful microprocessors. Initially, these were 8-bit machines, but soon 16-, 32-, and 64-bit CPUs became common. Many of these had the computing power of a mainframe computer, but for a fraction of the price. The amount of improvement that has occurred in computer technology in the past half century is truly staggering and totally unprecedented in other industries. From a machine that costs 100 million dollars and executed 1 instruction per second, we have come to machines that cost 1000 dollars and are able to execute 10 million instructions per second, a price/performance gain of 10^{12}.

The second development was the invention of high-speed computer networks. Local Area Networks (LANs) allow hundreds of machines, within a building to be connected in such a way that information can be transferred between machines in a few microseconds or so. Larger amounts of data can be moved between machines at rates of 10 to 1000 million bits/sec. Wide Area Networks (WANs) allow millions of machines over the globe to be connected at speeds varying from 64Kbps to gigabits per second.

The result of these technologies is that it is now not only feasible, but easy, to put together computing systems composed of large numbers of computers connected by high-speed network. And because concurrent events are taking place in today's high-performance computers due to the common practice of multiprogramming, multiprocessing, or multicomputing, change of computing paradigm from a single centralized system to that of Parallel and Distributed systems is inevitable.

Parallel Computing

Driven by the ever-increasing demand for higher performance, lower costs and sustained productivity in real-life applications, parallel processing has emerged as a key enabling technology in modern computers

Parallel Computing is the simultaneous execution of the same task (split up and specially adapted) on multiple or parallel processors in order to obtain faster results. There are many different kinds of parallel computers or "parallel processors". They are distinguished by the kind of interconnection between processors known as "processing elements" and memories.

One of the earlier attempts to classify parallel computing system is due to Flynn. His classification scheme is described in terms of two independent factors: the number of data streams that can be simultaneously processed, and the number of instruction streams that can be simultaneously processed. By “data stream” we mean the different pieces of data that are being operated upon at a given moment by the processors, while by “instruction stream” we mean the different instructions that are being executed simultaneously by the processors. According to Flynn’s classification, every computing falls into one of four categories: SISD, SIMD, MIMD and MISD:

Even though Flynn’s classification defines four names, only two of them are relevant to parallel computing – SIMD and MIMD. With SIMD (Single Instruction Multiple Data) the computing system is composed of a large number of identical processors that execute synchronously the same instruction; each one, however, operating on a different piece of data. Whereas in the case of MIMD (Multiple Instruction Multiple Data) the computing system consists of processors that are operating asynchronously, i.e., each processor may execute a different instruction while using a different piece of data. (See figures 2.1-1 and 2.1-2 below for complete description of SIMD and MIMD)

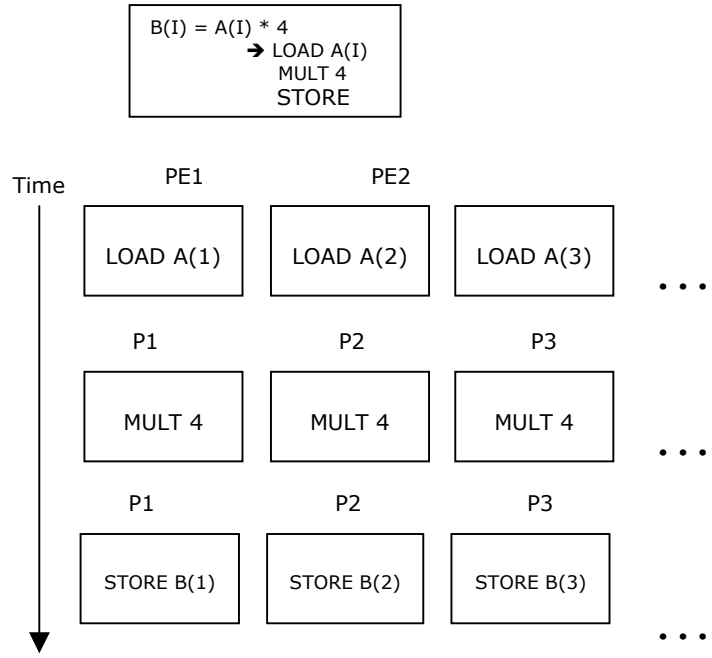

Figure 2.1-1: Flynn's SIMD Architecture

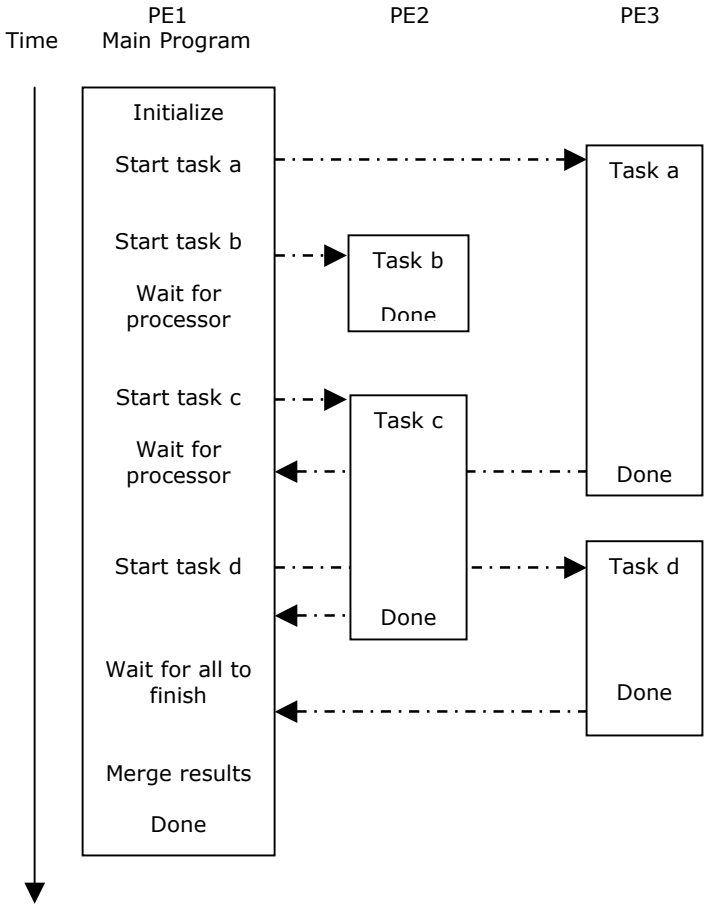

Figure 2.1-2: Flynn's MIMD Architecture

In the simplest sense, parallel computing is the simultaneous use of multiple processors to solve a computational problem. Processors to be used are usually of the same architecture, interconnected with high speed interconnection network, tightly coupled with memory, synchronized with single global clock and hence almost no room for flexibility, scalability and customizations. Distributed computing is the answer for this.

Distributed Computing

A Distributed System is a collection of independent computers that appears to its users as a single coherent system. This definition has two aspects. The first one deals with hardware: the machines are autonomous. One difference from parallel/cluster computing is that the computing machines are not homogenous in their architecture and could be located any where in the world. The second one deals with software: the users think they are dealing with a single system.

One important characteristic of distributed systems is that, differences between the various computers and the ways in which they communicate are hidden from users. But users and applications are also to interact with a distributed system in a consistent and uniform way, regardless of where and when interaction takes place.

Distributed systems should also be easy to expand or scale. This characteristic is a direct consequence of having independent computers, but at the same time, hiding how these computers actually take part in the system as a whole.

To support heterogeneous computers and networks while offering a single system view, distributed systems are often organized by means of a layer of software that is logically placed between a higher-level layer consisting of users and applications, and a layer underneath consisting of operating systems, as shown in Figure 2.1-3 below.

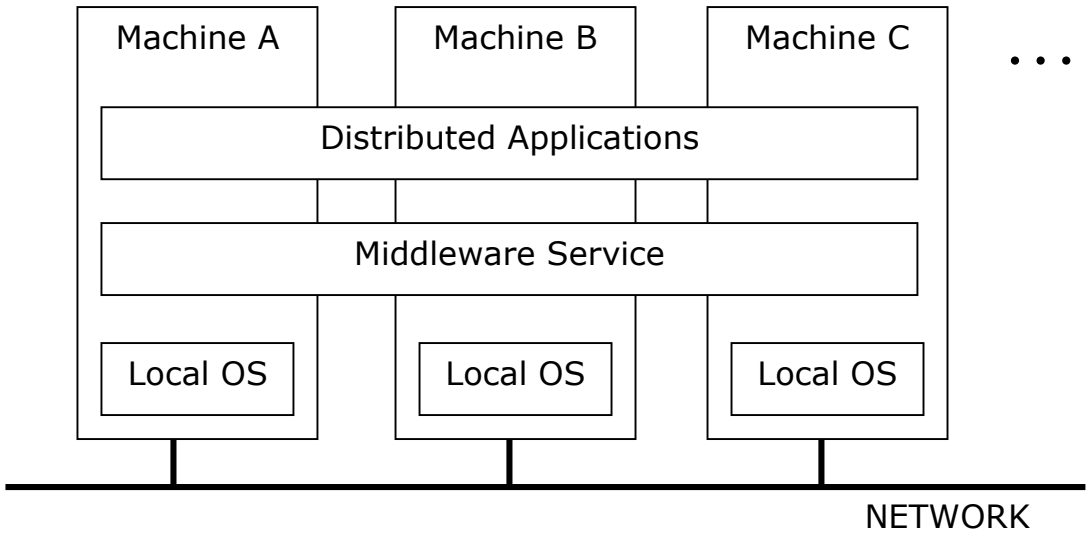

Figure 2.1-3: Distributed System organized as a middleware

Just because it is possible to build a distributed system does not necessarily mean that one has met and satisfied the computing needs. The main goal of a distributed system is to make it easy for users to access remote resources and share them with other users. While satisfying such needs, one should consider points like load balancing, scheduling, fault tolerance and etc…

2.2 Definitions of Grid Computing

Grid computing, the advanced stage of distributed computing, is an emerging new technology which is gaining much attention within the IT industry.

Because it is an emerging technology, Grid computing can mean different things to different people. A simple definition for Grid - a type of parallel and distributed system that enables the sharing, election and aggregation of geographically distributed computing resources depending on their availability, capability, cost, and user requirements for solving large-scale problems/applications.

The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. The sharing that we are concerned with is not primarily file exchange but rather direct access to

computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we can call a virtual organization (VO). [31]

The following are examples of VOs: the application service providers, storage service providers, cycle providers, and consultants who could be those engaged by a car manufacturer to perform scenario evaluation during planning for a new factory; members of an industrial consortium bidding on a new aircraft; a crisis management team and the databases and simulation systems that they use to plan a response to an emergency situation; and members of a large, international, multiyear high-energy physics collaboration. Each of these examples represents an approach to computing and problem solving based on collaboration in computation- and data-rich environments.

As these examples show, VOs vary tremendously in their purpose, scope, size, duration, structure, community, and sociology. Nevertheless, careful study of underlying technology requirements leads us to identify a broad set of common concerns and requirements. In particular, we see a need for highly flexible sharing relationships, ranging from client-server to peer-to-peer; for sophisticated and precise levels of control over how shared resources are used, including fine-grained and multi-stakeholder access control, delegation, and application of local and global policies; for sharing of varied resources, ranging from programs, files, and data to computers, sensors, and networks; and for diverse usage modes, ranging from single user to multi-user and from performance sensitive to cost-sensitive and hence embracing issues of quality of service, scheduling, co-allocation, and accounting.[31]

Current distributed computing technologies do not address the concerns and requirements listed above. For example, current Internet technologies address communication and information exchange among computers but do not provide integrated approaches to the coordinated use of resources at multiple sites for computation. Enterprise distributed

computing technologies such as CORBA and Enterprise Java enable resource sharing within a single organization. Storage service providers (SSPs) and application service providers (ASPs) allow organizations to outsource storage and computing requirements to other parties, but only in constrained ways: for example, SSP resources are typically linked to a customer via a virtual private network (VPN). Emerging "Distributed Computing" companies seek to harness idle computers on an international scale [29] but, to date, they support only highly centralized access to those resources. In summary, current technology either does not accommodate the range of resource types or does not provide the flexibility and control on sharing relationships needed to establish VOs.

It is here that Grid technologies enter the picture. Probably the most common description for Grid computing is the analogy to a power grid. When one plugs an appliance or other object requiring electrical power into a receptacle, he/she expects that there is power of the correct voltage available, but the actual source of that power is not known. The local power utility company provides the interface into a complex network of generators and power sources and provides with (in most cases) an acceptable quality of service for the energy demanded. Rather than each house or neighborhood having to obtain and maintain its own generator of electricity, the power grid infrastructure provides a virtual generator. The generator is highly reliable and adapts to the power needs of the consumers based on their demand.

The vision of Grid computing is similar. Once the proper kind of infrastructure is in place, a user will have access to a virtual computer that is reliable and adaptable to the user's needs. This virtual computer will consist of many diverse computing resources. But these individual resources will not be visible to the user, just as the consumer of electric power is unaware of how their electricity is being generated. To reach this vision, there must be standards for Grid computing that will allow a secure and robust infrastructure to be built. Standards such as the Open Grid Services Architecture (OGSA) and tools such as those provided by the GTK provide the necessary framework. Initially, businesses will build their own infrastructures (what one might call intra-Grids), but over time, these Grids will become interconnected. This interconnection will be made possible by standards such as OGSA and the analogy of Grid computing to the power grid will become real.

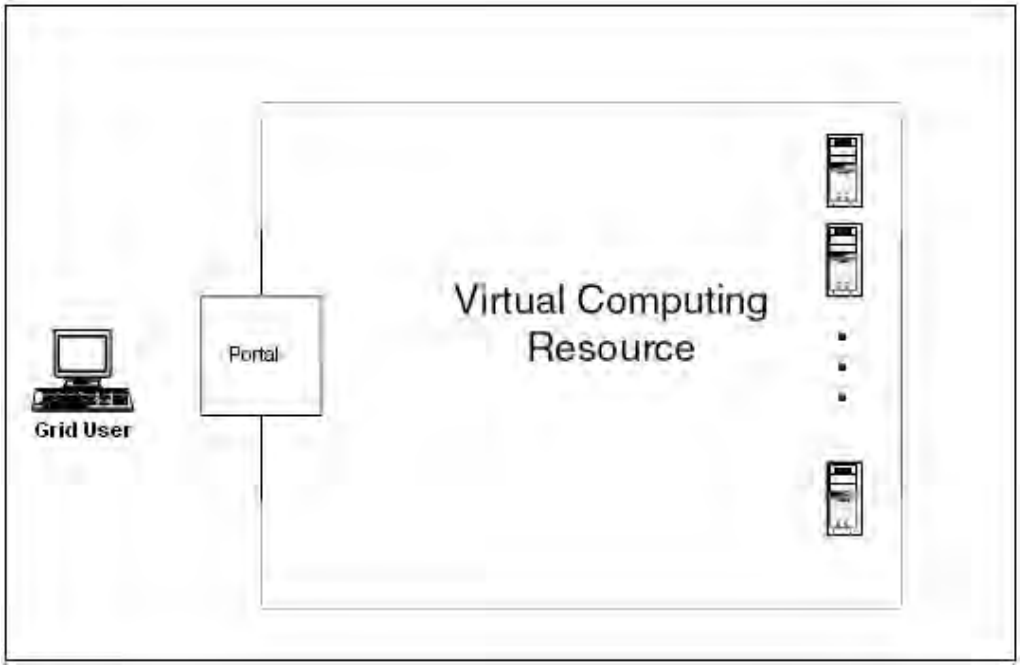

Figure 2.2-1: Possible user view of a Grid

Looking at the characteristics of Grid one can simply say, it is a computing environment which answers huge computational resource requirements of any type. But narrowing the requirements (or constructing a Grid for specialized purpose) will help a lot in obtaining the best out of the computing environment. Below is the discussion of different types of specialized Grids one can have.

2.2.1 Types of Grids

Grid computing can be used in a variety of ways to address various kinds of application requirements. Often, Grids are categorized by the type of solutions that they best address. The four primary types of Grids are summarized below. Of course, there are no hard boundaries between these Grid types and often Grids may be a combination of two or more of these. [16]

a) Computational Grid

A computational Grid focuses on setting aside resources specifically for computing power. In this type of Grid, most of the machines are high-performance servers.

b) Scavenging Grid

A scavenging Grid is most commonly used with large numbers of desktop machines. Machines are scavenged for available CPU cycles and other resources. Owners of the desktop machines are usually given control over when their resources are available to participate in the Grid.

c) Data Grid

A data Grid is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data. For example, we may have two universities doing life science research, each with unique data. A data Grid would allow them to share their data, manage the data, and manage security issues such as who has access to what data.

d) Network Grid

This is known as either a network 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.

2.3 Features of Grid Computing

When one deploys a Grid, it obviously is going to meet a set of users' computational requirements. To better match Grid computing capabilities to those requirements, it is useful to keep in mind the reasons for using Grid computing.

Exploiting underutilized resources

The easiest use of Grid computing is to run an existing application on a different machine. The machine on which the application is normally run might be unusually busy due to an unusual peak in activity. The job in question could be run on an idle machine elsewhere on the Grid.

There are at least two prerequisites for this scenario. First, the application must be executable remotely and without undue overhead. Second, the remote machine must meet any special hardware, software, or resource requirements imposed by the application.

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines are busy less than 5 percent of the time. In some organizations, even the server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage.

The processing resources are not the only ones that may be underutilized. Often, machines may have enormous unused disk drive capacity. Grid computing, more specifically, a "Data Grid", can be used to aggregate this unused storage into a much larger virtual data store, possibly configured to achieve improved performance and reliability over that of any single machine. If a batch job needs to read a large amount of data, this data could be automatically replicated at various strategic points in the Grid. Thus, if the job must be executed on a remote machine in the Grid, the data is already there and does not need to be moved to that remote point. This offers clear performance benefits. Also, such copies of data can be used as backups when the primary copies are damaged or unavailable.

Parallel CPU capacity

The potential for massive parallel CPU capacity is one of the most attractive features of a Grid. In addition to pure scientific needs, such computing power is driving a new evolution in industries such as the bio-medical field, financial modeling, motion picture animation, and many others. The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts. A CPU intensive Grid application can be thought of as many smaller "subjobs," each executing on a different machine in the Grid, to the extent that these subjobs do not need to communicate with each other, as the application becomes more "scalable". A perfectly scalable application will, for example, finish 10 times faster if it uses 10 times the number of processors.

Barriers often exist to perfect scalability. The first barrier depends on the algorithms used for splitting the application among many CPUs. In practice, linear speedup (i.e., speedup proportional to the number of processors) is very difficult to achieve. This is because many algorithms are essentially sequential in nature (Amdahl's law) [18].The second barrier appears if the parts are not completely independent; this can cause contention, which can limit scalability.

Virtual resources and virtual organizations for collaboration

Another important Grid computing contribution is enablement and simplification of collaboration among a wider audience. In the past, distributed computing promised this collaboration and achieved it to some extent. Grid computing takes these capabilities to an even wider audience, while offering important standards that enable very heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of virtual resources, as illustrated in figure 2.3-1 below. The users of the Grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger Grid.

Sharing starts with data in the form of files or databases. A “Data Grid” can expand data capabilities in several ways. First, files or databases can seamlessly span many systems and thus have larger capacities than on any single system. Such spanning can improve data transfer rates through the use of striping techniques. Data can be duplicated throughout the Grid to serve as a backup and can be hosted on or near the machines most likely to need the data, in conjunction with advanced scheduling techniques.

Sharing is not limited to files, but also includes many other resources, such as equipment, software, services, licenses, and others. These resources are “virtualized” to give them a more uniform interoperability among heterogeneous Grid participants.

The participants and users of the Grid can be members of several real and virtual organizations. The Grid can help in enforcing security rules among them and implement policies, which can resolve priorities for both resources and users.

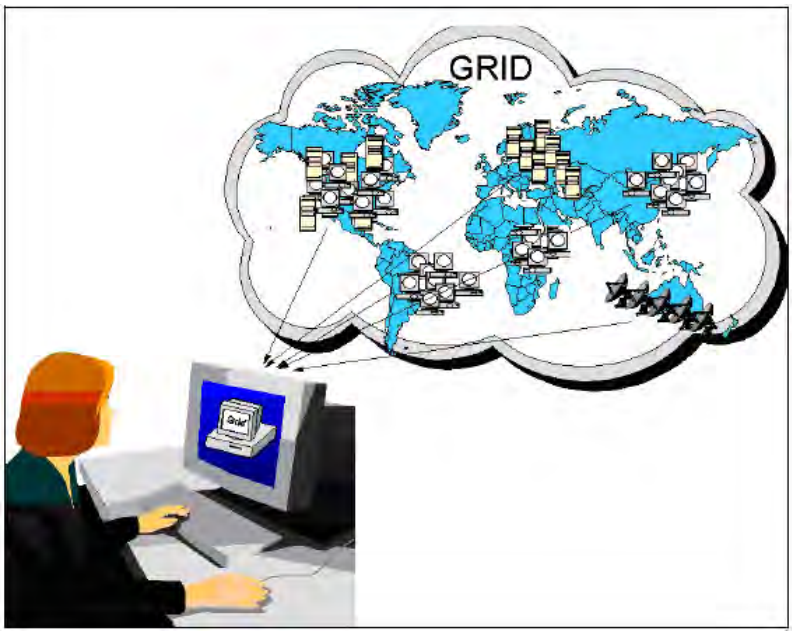

Figure 2.3-1: Grid virtualizing disperse resources (source IBM)

Reliability

High-end conventional computing systems use expensive hardware to increase reliability. They are built using chips with redundant circuits that vote on results, and contain much logic to achieve graceful recovery from an assortment of hardware failures. The machines also use duplicate processors with hot pluggability so that when they fail, one can be replaced without turning the other off. Power supplies and cooling systems are duplicated. The systems are operated on special power sources that can start generators if utility power is interrupted. All of this builds a reliable system, but at a great cost, due to the duplication of high-reliability components.

In the future, we will see a complementary approach to reliability that relies on software and hardware. A Grid is just the beginning of such technology. The systems in a Grid can be relatively inexpensive and geographically dispersed. Thus, if there is a power or other kind of failure at one location, the other parts of the Grid are not likely to be affected. Grid management software can automatically resubmit jobs to other machines on the Grid when a failure is detected. [16]

2.4 Resources in Grid

A Grid is a collection of machines, sometimes referred to as “nodes,” “resources,” “members,” “donors,” “clients,” “hosts,” “engines,” and many other such terms. They all contribute many combinations of resources to the Grid as a whole. Some resources may be used by all users of the Grid while others may have specific restrictions.

2.4.1 Computation

The most common resource is computing cycles provided by the processors of the machines on the Grid. The processors can vary in speed, architecture, software platform, and other associated factors, such as memory, storage, and connectivity. There are three primary ways to exploit the computation resources of a Grid. The first and simplest is to use it to run an existing application on an available machine on the Grid rather than locally. The second is to use an application designed to split its work in such a way that the separate parts can execute in parallel on different processors. The third is to run an application, which needs to be executed many times, on many different machines in the Grid.

2.4.2 Storage

The second most common resource used in a Grid is data storage. Each machine on the Grid usually provides some quantity of storage for Grid use, even if temporary. Storage can be memory attached to the processor or it can be “secondary storage” using hard disk drives or other permanent storage media.

Secondary storage in a Grid can be used in interesting ways to increase capacity, performance, sharing, and reliability of data. Many Grid systems use mountable networked file systems, such as Andrew File System (AFS), Network File System (NFS), Distributed File System (DFS), or General Parallel File System (GPFS). These offer varying degrees of performance, security features, and reliability features.

2.4.3 Communications

The rapid growth in communication capacity among machines today makes Grid computing practical, compared to the limited bandwidth available when distributed computing was first emerging. Therefore, it should not be a surprise that another important resource of a Grid is data communication capacity. This includes communications within the Grid and external to the Grid.

Communications within the Grid are important for sending jobs and their required data to points within the Grid. Some jobs require a large amount of data to be processed and it may not always reside on the machine running the job. The bandwidth available for such communications can often be a critical resource that can limit utilization of the Grid.

External communication access to the Internet, for example, can be valuable when building search engines. Machines on the Grid may have connections to the external Internet in addition to the connectivity among the Grid machines. When these connections do not share the same communication path, then they add to the total available bandwidth for accessing the Internet.

2.4.4 Software and Licenses

The Grid may have software installed that may be too expensive to install on every Grid machine. Using a Grid, the jobs requiring this software are sent to the particular machines on which this software happens to be installed. When the licensing fees are significant, this approach can save significant expenses for an organization.

3. GRID SOFTWARE ARCHITECTURE

Even though the Grid is in its early days, there is considerable consensus amongst the many developers of Grid technology about how it should be structured.

Just like civil engineers building a bridge, software engineers must specify an overall design for the structure they want to build; and of course before they start work on it. The design for the Grid software is called the "Grid Architecture". The Grid architecture identifies the fundamental components of the Grid, describes their purpose and function, and indicates how these components should interact with each other.

The Grid depends on underlying hardware, from the computers and communications networks that underlie the Grid, to the software for doing all sorts of complex calculations that will run on the Grid. Of all these components, though, the essence of the Grid - what really makes the whole thing possible - is the software that enables the user to access computers distributed over the network. This software is called "middleware", because it is distinct from the operating systems software that makes the computers run and also different from the applications software that solves a particular problem for a user.

The objective of the middleware is to get the applications to run on the right computers, wherever they may be on the Grid, in an efficient and reliable way. More generally speaking, the middleware's task is to organize and integrate the disparate computational resources of the Grid into a coherent whole.

The development of middleware is the main purpose of many of the Grid research and development projects currently underway around the globe and Globus Toolkit is the leader in this role.

3.1 Grid Software Components

This section presents some of the key Grid software components that must be discussed before dealing with a Grid computing architecture.

3.1.1 Management Software

Every Grid system must have some kind of management software. This software is used to keep track of resources available to the Grid and also to keep track of the different users on the Grid. This information is used primarily to decide where and to whom Grid jobs should be assigned.

Another job of the management software is measurement of the resources' capacity, utilization rate, traffic congestion and bottlenecks. Such information is used to determine the health of the Grid, alerting personnel to problems such as outages, congestion, or over commitment. This information is also used to determine overall usage patterns and statistics, as well as to log and account for usage of Grid resources.

3.1.2 Donor and Submission Software

Each machine contributing resources to the Grid must be enrolled as a member of the Grid and must have some software installed that allows the Grid to better manage and use its resources. The software installed on the donor machine can help to monitor its resources and also to send the resource information to the Grid management software. For example, in a "scavenging" Grid this information is used to inform the Grid management software of the availability of idle time in a machine.

Most importantly, the software installed on a donor machine can accept an executable job from the Grid management system and execute it. A user somewhere on the Grid submits a job for execution on the Grid. The Grid management software must communicate with the Grid donor software to send the job there. The donor Grid software must be able to receive the executable file or select the proper one from copies pre-installed on the donor machine. The software is executed and the output is sent back to the requester.

Usually any member machine of a Grid can be used to submit jobs to the Grid and initiate Grid queries. However, in some Grid systems, this function is implemented as a separate component installed on "submission nodes" or "submission clients." When a Grid is built using dedicated resources rather than scavenged resources, separate submission software is usually installed on the user's desktop or workstation.

3.1.3 Schedulers

Most Grid systems include some sort of job scheduling software. This software locates a machine on which to run a Grid job that has been submitted by a user. In the simplest cases, it may just blindly assign jobs in a round-robin fashion to the next machine matching the resource requirements. However, there are advantages to using a more advanced scheduler.

Some schedulers implement a job priority system. This is sometimes done by using several job queues, each with a different priority. As Grid machines become available to execute jobs, the jobs are taken from the highest priority queues first. Policies of various kinds are also implemented using schedulers. Policies can include various kinds of constraints on jobs, users, and resources. For example, there may be a policy that restricts Grid jobs from executing at certain times of the day.

3.1.4 Communications

A Grid system may include software to help jobs communicate with each other. For example, an application may split itself into a large number of subjobs. Each of these subjobs is a separate job in the Grid. However, the application may implement an algorithm that requires that subjobs communicate some information among them. The subjobs need to be able to locate other specific subjobs, establish a communications connection with them, and send the appropriate data. The open standard Message Passing Interface (MPI) and any of several variations is often included as part of the Grid system for just this kind of communication.

3.2 Architecture and Standards

In order to aggregate distributed and heterogeneous high end machines, standards are needed. The standardization of network communication between heterogeneous systems led to the explosion of the Internet. Similarly the emerging standardization for sharing resources will lead to the explosion of Grid computing.

3.2.1 Architecture

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

The components that are required to form a Grid can be divided into five layers based on their role in the Grid system. They are ***fabric***, ***connectivity***, ***resource***, ***collective*** and ***application*** layers interconnected as shown below in figure 3.2-1.

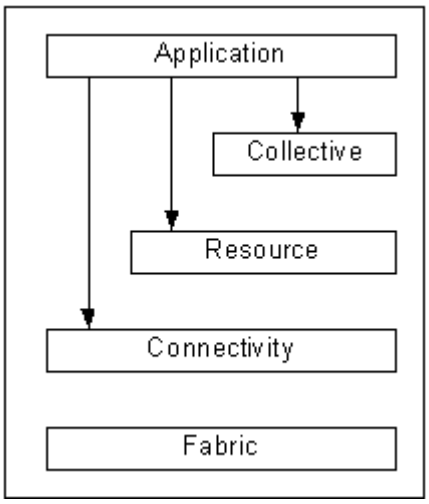

Figure 3.2-1: Architecture of Grid Protocol

Fabric

This is the lowest level of Grid architecture which consists of diverse computational resources such as computers, storage media, networks and etc…

Fabric components implement the local, resource-specific operations that occur on specific resources as a result of sharing operations at higher levels. While acting as an interface to local resources, the fabric may employ "local" protocols to achieve desired operation. If the existing "local" protocol lacks the required functionality, then the fabric is to fill in the gap.

Connectivity

The connectivity layer contains the core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between resources, whereas authentication protocols build on communication services are to provide cryptographically secure mechanisms for verifying the identity of users and resources.

The communication protocols within connectivity are to contain protocols for transport, routing, and naming. Whereas some of the requirements from authentication protocols are support for single sign on, delegation, interoperability with existing protocols and user based trust.

Resource

The resource layer contains protocols, Application Programmers Interfaces (APIs) and Software Developer Kits (SDKs) that exploit communication and authentication protocols for the secure negotiation, initiation, monitoring, control, and accounting of sharing operations on individual resources. Resource layer implementations of these protocols call Fabric layer functions to access and control local resources. The GTK is a commonly used source of connectivity, resource protocols and APIs.

Collective

The collective layer contains protocols, services, and APIs that implement interactions across collections of resources. The collective services are also based on protocols: ***information protocols***, which obtain information about the structure and state of the resources on the Grid, and ***management protocols*** which negotiate access to resources in a uniform way. Because they combine and exploit components from the relatively narrower resource and connectivity layers, the components of the collective layer can implement a wide variety of tasks without requiring new specific resource-layer components – and hence provide uniform access to resources.

Application

At the top of any Grid system are the user applications, which call on the components in other layers to complete their tasks. For example, a high-energy physics analysis application that needs to execute several thousands of independent tasks, each taking as input some set of files containing events, might proceed by

- obtaining necessary authentication credentials (connectivity layer protocols)
- querying an information system and replica catalog to determine availability of computers, storage systems, and networks, and the location of required input files (collective services)
- submitting requests to appropriate computers, storage systems, and networks to initiate computations, move data, and so forth (resource protocols) and
- monitoring the progress of the various computations and data transfers, notifying the user when all are completed, and detecting and responding to failure conditions (resource protocols).

Below is a figure showing the detailed interaction of Grid components (each implementing task or functionality defined by the Grid layer) while managing user application.

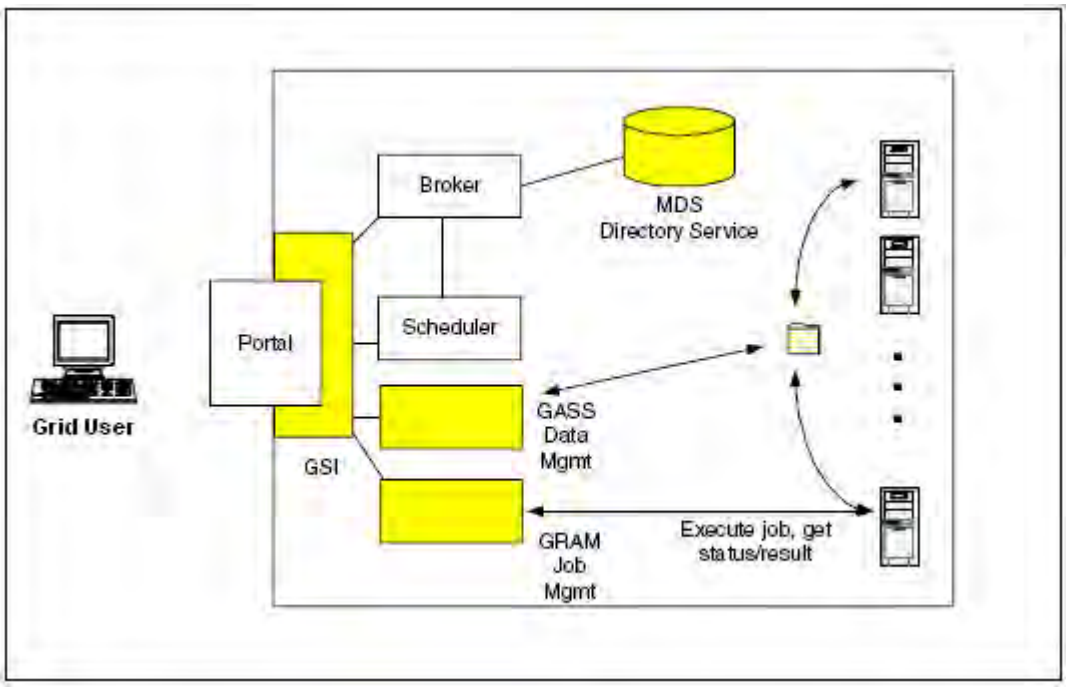

Figure 3.2-2: Interaction of Grid components (source IBM)

3.2.2 Standards

One big idea behind the Grid is open standards. The idea is to convince the community of software engineers currently developing the Grid, including those from major IT companies, to set common standards for the Grid up-front, so that applications made to run on one Grid will run on all others. This may seem idealistic - after all, many software companies make their profits precisely because they do not share their standards with others. However, because the very nature of the Grid is about sharing, it is generally perceived to be in everyone's self interest to set common, open standards.

Grid-specific standards are currently being developed by the Global Grid Forum (GGF), an open community initiative that aims, as stated in its Web page, “to promote and support the development, deployment, and implementation of Grid technologies and applications via the creation and documentation of ‘best practices’ technical specifications, user experiences, and implementation guidelines.”[6]

Two main standards which are the results of the GGF initiative, Open Grid Services Architecture (OGSA) and Open Grid Services Infrastructure (OGSI), are seen as the key

reference for future Grid development projects. OGSI's and OGSA's goals are to provide the framework to support the creation of set of searchable services, callable by any system on a Grid.

3.2.2.1 Open Grid Services Architecture (OGSA)

This is an evolving standard with significant industry support. It defines the blue print of what the Grid services are, what they should be capable of, what type of technologies they should be based on, but doesn't give a technical and detailed specification for the implementation.

OGSA also defines the programming model of Grid services. It provides instructions on how to build a Grid service, by outlining the required components needed to build and deliver an enterprise-class Grid solution.

As described by GGF, successful realization of OGSA requires the early definition of core services, behaviors, resource models, bindings and so forth: which are assumed to be basics for the construction of OGSA platform. The OGSA working group within the Global Grid Forum has been formed to define this OGSA Platform by

a) specifying, in broad but somewhat detailed terms, the scope of important services required to support both e-science and e-business applications,
b) identifying a core set of such services that are viewed as essential for many Grid systems and applications, and
c) specifying, at a high-level, the functionalities required for these core services and the interrelationships among those core services.

3.2.2.2 Open Grid Services Infrastructure (OGSI)

OGSI is the concrete specification of OGSA infrastructure. Based on the specification of OGSI, OGSA has provided a mechanism for integrating key Grid technologies with web service mechanisms so that a distributed system framework can be created.

OGSI is the middleware, the "Java 2 Platform" for Grid services. It defines how to build a Grid service, outlining the mechanisms for creating, managing, and exchanging information for Grid services.

According to GGF, building on both Grid and Web services technologies; the OGSI defines mechanisms for creating, managing, and exchanging information among entities called Grid services. Concisely, a Grid service is a Web service that conforms to a set of conventions (interfaces and behaviors) that define how a client interacts with a Grid service. These conventions, and other OGSI mechanisms associated with Grid service creation and discovery, provide for the controlled, fault resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications.

The Grid services defined by OGSA and OGSI are extensions of web services [35]. The GTK (version 3) is implemented based on the specification of OGSI standards and hence OGSA standards. Figure 3.2-3 presents the interrelationship between OGSA, OGSI, Web Services and the GTK implementation.

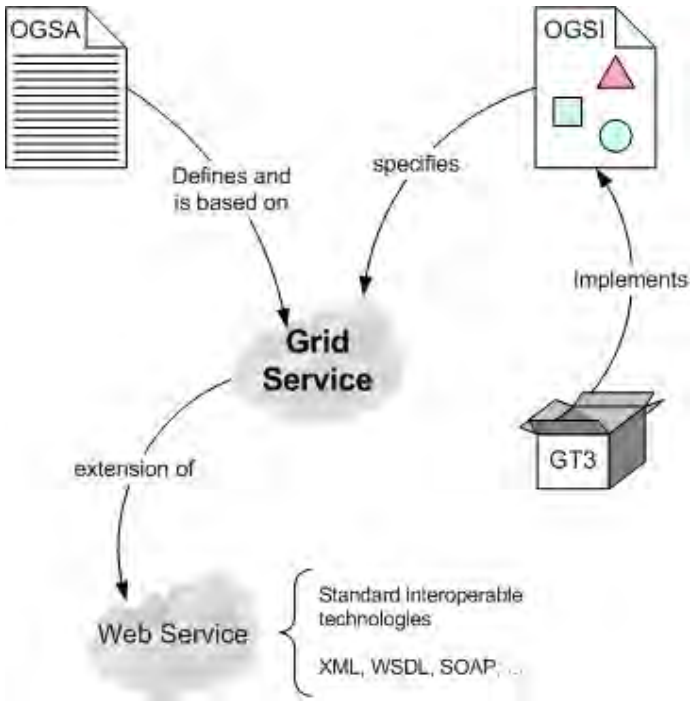

Figure 3.2-3: Major players in the Grid Services (source Borja Sotomayor)

3.3 Globus Toolkit (GTK)

This is an open source toolkit released by the Globus Alliance whose members include researchers at Argonne National Laboratory, the University of Southern California's Information Sciences Institute, the University of Chicago, the University of Edinburgh, and the Swedish Royal Institute of Technology. Some of the prominent corporate sponsors of the alliance include IBM, Microsoft and Cisco.

The GTK is a fundamental enabling technology for the Grid, letting people share, computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security and file management.

The main components of the GTK are security, resource management, information services, and data management.

3.3.1 Security

Security is an important component in the Grid computing environment. The GTK, at its base, has the Grid Security Infrastructure (GSI), which provides many facilities to help manage the security requirements of the Grid environment.

Grid security infrastructure (GSI)

The GSI within the GTK is responsible for the authentication, authorization, and secure communication between Grid resources.

a) **Authentication**

GSI contains the infrastructure and facilities to provide a single sign-on environment. Through the grid-proxy-init command or its related APIs, a temporary proxy is created based on the user's private key. This proxy provides authentication and can be used to generate trusted sessions and allow a server to decide on the user's authorization.

A proxy must be created before a user can submit a job to be run or transfer data through the GTK facilities. Depending on the configuration of the environment, a proxy may or may not be required to query the information services database.

b) Authorization

Authentication is just one piece of the security puzzle. The next step is authorization. That is, once a user has been authenticated to the Grid, the task of authorization is to decide on what the user is privileged to do.

In GSI this is handled by mapping the authenticated user to a local user on the system where a request has been received. The proxy passed by the operation request (such as a request to run a job) contains a distinguished name of the authenticated user. And a file on the receiving system is used to map this distinguished name to a local user so that executions can be possible.

Through this mechanism, either every user of the Grid could have a user ID on each system within the Grid (which would be difficult to administer if the number of systems in the Grid becomes large and changes often), or users could be assigned to virtual groups. For example, all authenticated users from a particular domain may be mapped to run under a common user ID on a particular resource. This helps separate the individual user ID administration for clients from the user administration that must be performed on the various resources that make up the Grid.

c) Grid secure communication

It is important to understand the communication functions within the GTK. By default, the underlying communication is based on the mutual authentication of digital certificates, Secure Socket Layer (SSL) and Transport Layer Security (TLS).

To allow secure communication within the Grid, the OpenSSL package is installed as part of the GTK. It is used to create an encrypted tunnel using SSL/TLS between Grid clients and servers.

The digital certificates that have been installed on the Grid computers provide the mutual authentication between the two parties. The SSL/TLS functions that OpenSSL provides will encrypt all data transferred between Grid systems. These two functions together provide the basic security services of authentication and confidentiality.

3.3.2 Resource Management

The Grid resource manager is concerned with resource assignments as jobs are submitted. It acts as an abstract interface to the heterogeneous resources of the Grid. The resource management component provides the facilities to allocate a job to a particular resource, provides a means to track the status of the job while it is running and its completion information, and provides the capability to cancel a job or otherwise manage it.

In Globus, the remote job submission is handled by the Globus Resource Allocation Manager (GRAM).

Globus Resource Allocation Manager (GRAM)

When a job is submitted by a client, the request is sent to the remote host and handled by a gatekeeper daemon. The gatekeeper creates a job manager to start and monitor the job. When the job is finished, the job manager sends the status information back to the client and terminates.

The GRAM subsystem consists of the following elements:

- The globusrun command and associated APIs – these are used to submit a job with a Grid resource and typically passed as an RSL string
- Resource Specification Language (RSL) – this is a language used by clients to specify the job to be run
- The gatekeeper daemon – daemon, similar to that of inetd process of Unix Operating System (in terms of functionality), which provides secure communication between clients and servers

- The job manager – daemon created by the gatekeeper daemon, as part of the job requesting process, in order to provide interfaces that control allocation of each local resource
- Dynamically-Updated Request Online Coallocator (DUROC) – an API which allows users to submit multiple jobs to multiple GRAMs with one command

3.3.3 Information Services

Information service is a vital component of the Grid infrastructure. It maintains knowledge about resource availability, capacity, and current utilization. Within any Grid, both CPU and data resources will fluctuate, depending on their availability to process and share data. As resources become free within the Grid, they can update their status within the Grid information services. The client, broker, and/or Grid resource manager uses this information to make informed decisions on resource assignments.

The Grid Information Service (GIS), also known as the Monitoring and Discovery Service (MDS), provides the information services in Globus. The MDS uses the Lightweight Directory Access Protocol (LDAP) as an interface to the resource information.

Examples of information to be stored in LDAP include:

- Static host information - Operating system name and version, processor vendor/model/version/speed/cache size, number of processors, total physical memory, total virtual memory, devices, service type/protocol/port
- Dynamic host information - Load average, queue entries, and so on
- Storage system information - Total disk space, free disk space, and so on
- Network information - Network bandwidth, latency, measured and predicted
- Highly dynamic information - Free physical memory, free virtual memory, free number of processors, and so on

MDS provides access to static and dynamic information of resources. Basically, it contains the following components:

- Grid Resource Information Service (GRIS) - GRIS is the repository of local resource information derived from information providers.
- Grid Index Information Service (GIIS) - GIIS is the repository that contains indexes of resource information registered by the GRIS and other GIISs.
- Information providers - The information providers translate the properties and status of local resources to the format defined in the schema and configuration files.
- MDS client - The MDS client is based on the LDAP client command, **ldapsearch**, or an equivalent API. A search for information about resources in the Grid environment is initially performed by the MDS client.

3.3.4 Data Management

When building a Grid, the most important asset within the Grid is data to be processed. In the design, one has to determine the data requirements and how to move data around the Grid infrastructure or otherwise how to access the required data in a secure and efficient manner.

Globus provides the GridFTP and Global Access to Secondary Storage (GASS) data transfer utilities in the Grid environment. In addition, a replica management capability is provided to help manage and access replicas of a data set.

GridFTP

The GridFTP facility provides secure and reliable data transfer between Grid hosts. Its protocol extends the File Transfer Protocol (FTP) to provide additional features such as support for GSI and Kerberos authentication system, parallel and striped data transfer.

Global Access to Secondary Storage (GASS)

GASS is used to transfer files between the GRAM client and the GRAM server. GASS also provides libraries and utilities for the opening, closing, and pre-fetching of data from datasets in the Globus environment. A cache management API is also provided.

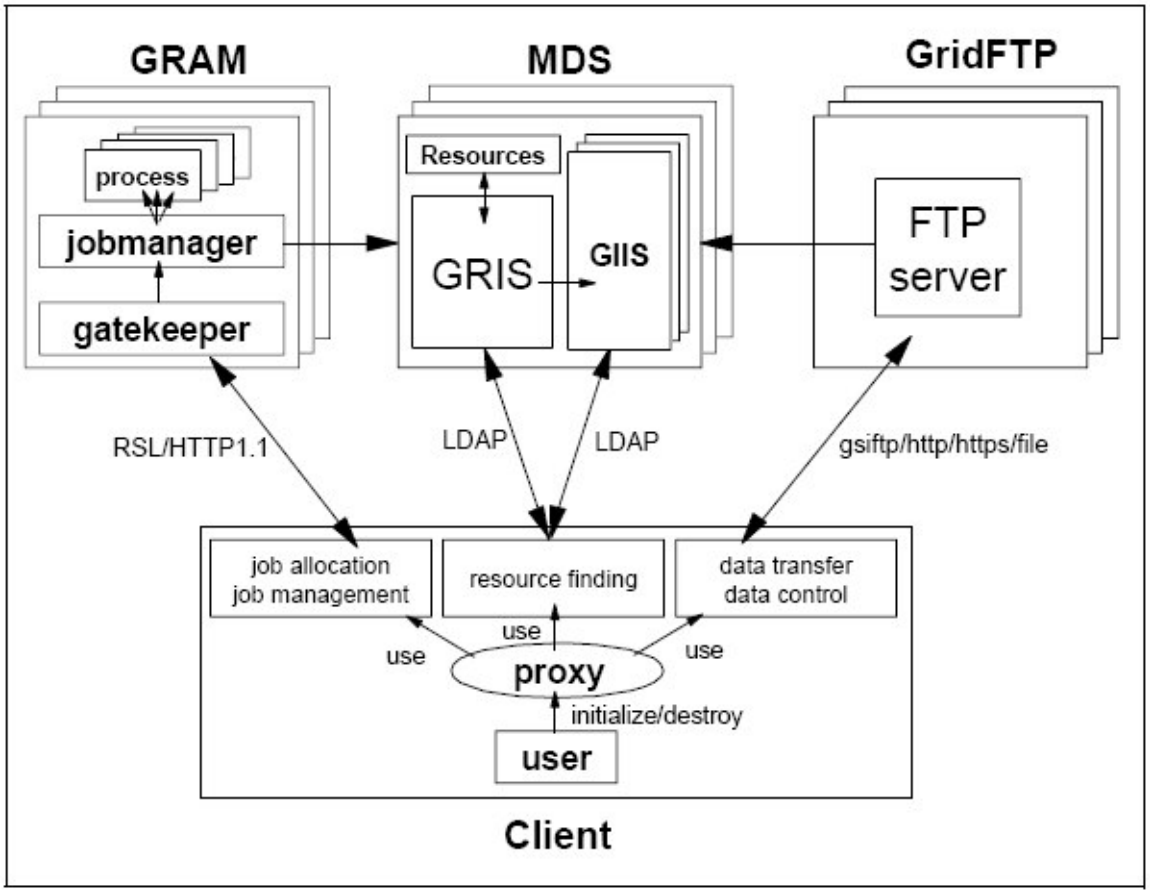

Figure 3.3-1: The system overview of Globus Toolkit (source IBM)

4. TESTBED DESIGN AND DEPLOYMENT

This section gives an overview of the design and configuration of the software and hardware used for the testbed deployment. Details of the configurations and installations are provided in Appendix A of this paper. The testbed environment, constructed with Globus Toolkit version 3 (GTK3), consists of 5 Intel architecture machines out of which three of them having RedHat 9.0 and the remaining two installed with SuSE 8.1.

4.1 Design

The design and implementation of the Gird testbed has followed an incremental design approach. But before going to the actual deployment, a careful analysis and planning of the resources and the topology are performed as presented in the subsequent sections.

4.1.1 System Requirements

Requirements for GTK installation can be classified into two categories: hardware and software.

Hardware Requirement

According to sources from IBM [16], Globus Toolkit can be installed with minimal hardware requirements of at least 133MHz Pentium processor and 16MB of RAM. But since the software installed needs testing by running sample applications, it is wise to use computers with speed greater than 133MHz and memory greater than 16MB.

The specific hardware resources used for the testbed deployment are

- 100Mbps LAN
- 3 Dell Optiplex GX280 computers each with 256MB of RAM, 750MHz of processor speed and 12.0 GB of HDD
- 1 Pentium IV Del Optiplex GX240 computer with 1.80GHz processor speed, 262 MB of memory and 9.0GB of HDD

- 1 Dell Latitude C840 Laptop with a memory of 256MB, 1.80 PIV processor speed and 10.0GB HDD

Software Requirement

GTK3 needs several files, or tools, in order to complete the installation. Depending on the type of environment some tools may not be necessary; as some operating systems are to have some of these required packages. Below are basic requirements for deploying the testbed.

- Operating system (different flavors of Linux, MS Windows is also possible)
- Globus Toolkit
- J2SE 1.4.2+ SDK.
- Ant 1.5.1+ (1.6.1+ if using Java 1.5).
- C compiler (gcc)
- GNU tar
- GNU sed
- zlib 1.1.4+
- GNU Make
- sudo
- JDBC compliant database

It is good to download and collect all the required softwares under a single installation source directory before starting installation. And all these softwares should be installed on the Grid servers.

4.1.2 Design Topology

The engineering activities in a system development cycle produce models and making models is a large part of an engineer's work. The testbed design model depicted in figure 4.1-1 is a model which best simulates the interconnection scenario of the three campuses of AAU.

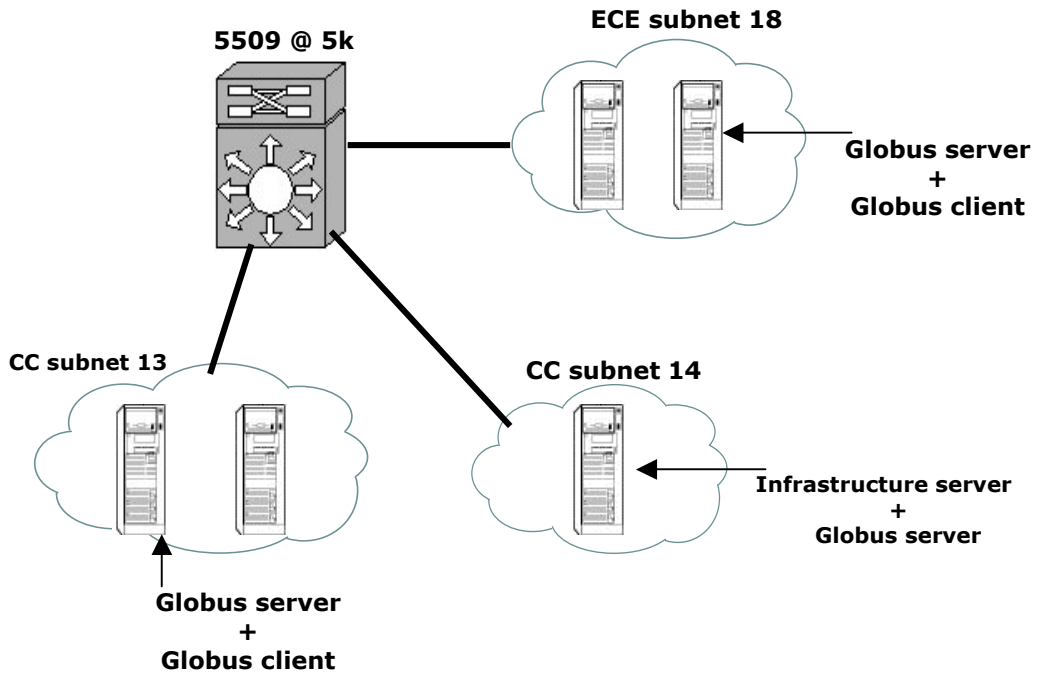

Figure 4.1-1: Testbed Design Model

Though physically all the machines are connected to the core switch of Technology Campus, their logical connection is distributed to the different subnets of AAU. And by following an incremental design approach, one can extend the size of the Grid to any possible number of interconnected resources that can be afforded.

In addition to the design model presented above, the following attributes, which are very important for workings of the Grid, are planned carefully.

Naming and Addressing

Table 4-1 summarizes the names of the machines to be used in the Grid, their IP addresses and the software to be installed on them.

Table 4-1: Name of machines used in the Grid

Hostname	IP Address	Description
alpha.grid.aau.edu.et	10.5.14.6	NFS, NTP, CA and Globus Server
beta.grid.aau.edu.et	10.5.14.7	Globus Server
gamma.grid.aau.edu.et	10.5.14.8	Globus Server
omega.grid.aau.edu.et	10.5.15.50	Globus Server
laptop.grid.aau.edu.et	10.5.13.50	Globus Server

Users and Groups

Table 4-2: List of users used in the Grid

User ID	Group ID	Activities
Root	root	Super user needs
Globus	globus	For installation and execution of GTK
eceGrid	eceGrid	For jobs execution on the Grid.

Directories

Table 4-3 presents the Globus directories used for the testbed deployment.

Table 4-3: Directories used for the Globus installation

Directory	Ownership	Description
/usr/local/globus	globus:globus	Base directory of Globus
/home/globus/install	globus:globus	Installation directory

Certificate

Table 4-4, below, describes the certificate naming conventions used for the testbed deployment.

Table 4-4: Naming conventions used for CA setup

Tag	Description	Value
C	Country Name (2 letter code)	C=et
ST	State or Province Name	ST=Addis
L	Locality Name (e.g. city)	L=Addis
O	Organization Name (e.g. company)	O=AAU
OU	Organization Unit Name (e.g. section)	OU=ECE
CN	Common Name (e.g. server name)	CN=alpha
Email	E-mail address of the CA administrator	Email=abyot@ece.aau.edu.et

4.2 Deployment

Deploying GTK is the very complex and difficult part of this thesis work. Deployment becomes complex because the middleware package of Grid is to contain a number of different packages which by themselves contain or depend on third party software packages.

As mentioned in chapter three of this paper, Globus is to contain at least the following four major parts:

1. Security
2. Resource Manager
3. Information Service and
4. Data Manager

Therefore, deployment of a Gird testbed is to include the proper installation and configuration of each of these four major software components. Below is an overview of the major workings in installing these software components with detailed discussions and documentation of the actual work in Appendix A of this paper.

Globus installation starts with first collecting, installing and configuring the prerequisite softwares described in the README file of Globus software. Of these requirements, Java SDK, Apache Ant, Junit and any Database Software (preferably UNIX compatible) are the major ones. Because all these requirements are to be installed on each and every machine, preparing a central repository machine is economical (instead of storing on each and every machine) and of great help. For this purpose, one of the machines in the testbed is selected and made to act as a file server by installing an NFS server on to it. Once the requirements are collected, their installation is performed easily as outlined in their README (or INSTALL) files.

One very important task to be considered in Grid is the issue of timing. This is very important as the Globus is to depend on certificates in order to secure resources in the Grid, and these certificates are controlled by digital signatures and expiry times. As a result of this, certificates granted for a fixed period of time need to be checked for their validity and expiration and hence the need to have time synchronization or common clock between the machines in the Grid. All the machines in the Grid are synchronized by reading their time from a centrally selected, installed and configured Network Time Protocol (NTP) server.

Security is an important component in the Grid computing environment; and this is achieved through GSI of Globus. And at the core of GSI we have certificates and their issuer – Certificate Authority (CA). From version 3.x and up, a simple CA package is included in the releases of Globus; which according to developers of Globus "a service that issues low-quality certificates". For this reason, an independent CA is installed and configured instead of the simple CA package of Globus.

Following the prerequisite installations and configurations is the installation and configuration of Globus package itself, and this installation is performed according to the README (or INSTALL) files of Globus and expertise of Linux for a number of encounters and customizations; which arise mainly due to the effect of surroundings and platform dependencies.

The complete procedure of deployment is in Appendix A of this paper.

5. TESTING AND RESULTS

Although a number of problems were faced by this thesis work during its course, that didn't stop the researcher to make the best out of the available resources and administration setups in the AAU. As was stated in the initial proposal of this thesis, one of the intentions was to setup a Grid testbed by using few machines of AAU from its three main campuses; science, technology and main campus. But, due to limitations in time, resource (both human and computing facilities for research) and as well as administrative issues, the testbed deployment was not able to consider machines from the different campuses. Instead the researcher outlines a best possible scenario which simulates the possibility of using resources from different campuses by using few resources from technology campus, specifically from the Department of Electrical and Computer Engineering.

As presented in the design model, the testbed resources from the Department of Electrical and Computer Engineering are partitioned to belong to different networks (by applying the concept of Virtual LANs and subnets) so that the intentions of the thesis can be met other way round. As a result of this, the testbed deployed is a suitable simulation model of AAU's large scale Grid deployment.

After deploying the Grid testbed properly in place, the next action performed, by the researcher of this thesis, was the testing of different resources with in the Grid for their functionality and performance.

Below are discussions of the sample test demo application that was executed on the testbed. In the discussions, an overview of the test application, its characteristics, considerations for Grid as well as its benefits from the testbed are presented.

5.1 Test Application

As stated earlier, Grid computing is the utilization of available computing resources across networks and hence providing immense computing power for user applications. Answering applications' computational resource requirement doesn't mean that any kind of application

can be executed on a Grid. Instead it is those applications with profiles listed below which can best benefit from Grid technology. The profiles are

- Applications taking too long to finish.
- CPU intensive applications.
- Storage intensive applications.
- Geographically distributed applications.

For testing purpose, an application with typical characteristics of large CPU and memory requirements is selected. The selected application, RainbowCrack, is an efficient method of cracking Windows LM passwords. The method, known as the Faster Time-Memory Trade-Off Technique [31], uses pre-calculated tables consisting of every possible combination of characters in a Windows password and a sophisticated search algorithm.

5.1.1 LM Hash

LM Hash or LanManager Hash is a format that Microsoft Windows uses to store Windows user passwords that are less than 15 characters in length. This type of hash was the only type of encryption used in early versions of Windows (up to Windows Me) and is still supported in recent versions for backward compatibility.

A password is a security measure used to restrict logon names to user accounts and access to computer systems and resources. A password can be made up of letters, numbers, and symbols; passwords can also be blank. It is usually recommended to use complex passwords to help ensure that these passwords provide the best security possible. These complex passwords are much more resistant to attack than blank or simple passwords.

Instead of storing the user account password in clear-text, Windows generates and stores user account passwords by using two different password representations, generally known as hashes. When one sets or changes the password for a user account to a password that contains fewer than 15 characters, Windows generates both LM Hash and a Windows NT hash (NTLM Hash) of the password. These hashes are stored in the local Security Accounts Manager (SAM) database or in Active Directory.

The LM Hash is relatively weak compared to the NTLM Hash, but it is needed for backward compatibility with Windows 9x clients, and used, typically, to authorize remote connection to a given machine. To generate the LM Hash, the system converts the password from UNICODE to ANSI (one byte per character), and translates all characters into uppercase. After that, the password is divided to two chunks (7 chars each, padded with zeros if needed). Each part is used as a Data Encryption Standard (DES) encryption key, to encrypt the pre-defined constant, and the results of encryption are stored in the system (merged into a single 16-byte value). So, if a system uses LM authentication (and so LM Hashes are available), the real password length or complexity is just 7 characters, and the 14-character password is not much stronger than one of 7 characters.

Because of the security weaknesses inherent in LM encryption, Microsoft is pushing for the replacement of LM by NTLM algorithm, and it claims that support for LM will be completely eliminated in the new Windows Vista operating system.

5.1.2 Cracking Techniques

Password cracking is the process of recovering secret passwords from hashed data that has been stored in or transmitted by a computer system. The purpose of password cracking might be to help a user recover a forgotten password, to gain unauthorized access to a system, or as a preventive measure by the system administrator to check for easily crackable passwords. On top of these, it is a profession by itself.

Techniques used for cracking include guessing, dictionary attack, brute force attack and pre-computation.

Guessing

Not surprisingly, many users choose weak passwords, usually one related to themselves in some way. It may be: blank, the word 'password', the user's name or login name, the user’s birthplace or date of birth and so on.

The determined cracker can easily develop a computer program that accepts personal information about the user being attacked and generates common variations for passwords suggested by that information.

Dictionary attack

A dictionary attack uses a file of dictionary words. The cracking program encrypts each word in the dictionary using the same algorithm used to create the hash to be cracked; it then compares the two hashes, the newly generated and the one to be cracked.

Guessing, combined with dictionary attacks, have been repeatedly and consistently demonstrated for several decades to be sufficient to crack perhaps as many as 50% of all account passwords on production systems.

Brute force attack

A brute-force attack simply tries every possible combination of characters until a match is found. Given enough time, a brute-force attack can deduce any password.

Pre-computation

In its most basic form, pre-computation involves hashing each word in the dictionary and storing the <plaintext, hash> pairs in a way that enables lookup on the hash field. This way, when a new encrypted password is obtained, password recovery is instantaneous. Pre-computation can be very useful for a dictionary attack if salt is not used properly, and the dramatic decrease in the cost of mass storage has made it practical for fairly large dictionaries. RainbowCrack is typical implementation of pre-computation cracking technique

5.1.3 RainbowCrack

RainbowCrack, full package from http://www.antsight.com/zsl/rainbowcrack/, is a general propose implementation of Philippe Oechslin's faster time-memory trade-off technique. [26]

In short, the RainbowCrack tool is a hash cracker. A traditional brute force cracker tries all possible plaintexts one by one in cracking time. It is time consuming to break complex password in this way. The idea of time-memory trade-off is to do all cracking time computation in advance and store the result in files so called "rainbow table". It does take a long time to pre-compute the tables. But once the one time table pre-computation is finished, a time-memory trade-off cracker can be hundreds of times faster than a brute force cracker.

Modern desktop machines can crack alphanumeric LM Hashes in hours with a brute force attack or in a few minutes using the RainbowCrack cryptanalysis attack.

5.1.4 Testing

The testing part of this thesis is nothing but running LM Hash cracker over the Grid testbed. And running the RainbowCrack application requires four steps to be performed. These are:

1. Selecting configuration
2. Table pre-computation
3. Sorting and
4. Cracking

Selecting Configuration

Because RainbowCrack is a general purpose hash cracking tool, it is necessary to first decide on the configuration of the attack. As presented by Philippe Oechslin on his “faster time-memory trade-off technique” paper, there are a number of parameters to be adjusted: the success rate, the hash type to work with, the charset to use, the available memory, and so on.

To minimize the size of the hash table, as well as the time of table pre-computation, the charset configuration is restricted to contain LM hashes of alphabets “ABCDEFGHIJKLMNOPQRSTUVWXYZ” only. And the specific configuration selected is called “lm configuration 0” with attributes listed in Table 5-1.

Table 5-1: RainbowCrak's lm 0 configuration attributes

Attribute	Value
charset	ABCDEFGHIJKLMNOPQRSTUVWXYZ
keyspace	8353082582
Success probability	0.9990
Table size	610MB
Hash type	Lm

Table pre-computation

Generating an LM hash table for the alphabets is a very critical part of the RainbowCrack application, especially in the context of Grid environment. Looking at the concept of time-memory trade-off, RainbowCrack is to do the actual cracking by walking through a huge pre-computed "rainbow table". In other words, cracking by RainbowCrack is nothing but searching inside a huge table. And hence it is advantageous to divide the huge table into smaller pieces and then run the searches parallely on to the Grid machines.

There is a utility called "rtgen.exe" in the RainbowCrack package for hash table generation. Running

```
rtgen lm alpha 1 7 2100 8000000 all
```

starts the generation of the required rainbow table of size 610MB. It takes about 70 hours to complete on an average desktop machine. So running the following commands on the five machines of the Grid testbed generates 5 different tables of each 128MB with an approximate running time of 13.4 hours.[26][21]

```
alpha#  rtgen lm alpha 1 7 0 2100 8000000 all
beta#   rtgen lm alpha 1 7 1 2100 8000000 all
gama#   rtgen lm alpha 1 7 2 2100 8000000 all
omega#  rtgen lm alpha 1 7 3 2100 8000000 all
laptop# rtgen lm alpha 1 7 4 2100 8000000 all
```

Dividing the single table into pieces and then distributing it across the Grid has solved the need for a huge memory requirement from a single machine (other configurations of RainbowCrack require free disk space in the range of 100GB).

After the tables are distributed on the testbed machines, each testbed machine can start searching (cracking) independently.

Sorting

Searching can be very efficient if applied with sorting. RainbowCrack package has a tool called "rtsort.exe" which can take a rainbow table (hash file) as an input and then provide a sorted version of it. The following are commands used on the testbed machines; each one working on its own table for sorting.

```
alpha#      rtsort      lm_alpha#1-7_0_2100x8000000_all.rt
beta#       rtsort      lm_alpha#1-7_1_2100x8000000_all.rt
gama#       rtsort      lm_alpha#1-7_2_2100x8000000_all.rt
omega#      rtsort      lm_alpha#1-7_3_2100x8000000_all.rt
laptop#     rtsort      lm_alpha#1-7_4_2100x8000000_all.rt
```

Cracking

Cracking is the final step in running the RainbowCrack application. And one final task performed in configuring the application for the testbed is, preparation of a wrapper script containing the actual rainbow cracker; "rcrack.exe". This script, shown below, is to launch the rainbow cracker through GTK resource manager – globusrun.

```
for h in laptop omega alpha beta gama;
do globusrun -o -r $h '&(executable=rcrack)\
      (arguments=*.rt –l "Hash to be cracked")';
done;
```

In order for this script to work, the ***"Hash to be cracked"*** need to be first written in a file. And it is this file that should be passed as a value for the "-l" attribute in the above script.

Below are sample outputs of the testbed machines while executing the RainbowCrack application in order to crack the windows LM Hash SAM database containing test username ***"abyot"*** with password ***"ECEMAIL"***.

```
Machine laptop using Rainbow Table 0

        lm_alpha#1-7_0_2100x8000000_all.rt:
        128000000 bytes read, disk access time: 4.02 s
        verifying the file...
        searching for 10 hashes...
        cryptanalysis time: 48.30 s

        statistics
        ---------------------------------------------------
        plaintext found:           0 of 10 (0.00%)
        total disk access time:    4.02 s
        total cryptanalysis time: 48.30 s
        total chain walk step:     22018510
        total false alarm:         21243
        total chain walk step due to false alarm: 14759331

        result
        ---------------------------------------------------
        abyot           <notfound>  hex:<notfound>
        Administrator   <notfound>  hex:<notfound>
        Guest             hex:
        IUSR_GRIDS                                  <notfound><notfound>
        hex:<notfound><notfound>
        IWAM_GRIDS                                  <notfound><notfound>
        hex:<notfound><notfound>
        TsInternetUser                              <notfound><notfound>
        hex:<notfound><notfound>
        VUSR_GRIDS                                  <notfound><notfound>
        hex:<notfound><notfound>
        __vmware_user__     hex:

Machine omega using Rainbow Table 1

        lm_alpha#1-7_1_2100x8000000_all.rt:
        128000000 bytes read, disk access time: 6.20 s
        verifying the file...
        searching for 10 hashes...
        plaintext of d90317151676e564 is ECEMAIL
        plaintext of 52567951f8073282 is ELMQPPS
        cryptanalysis time: 22.75 s

        statistics
        ---------------------------------------------------
        plaintext found:           2 of 10 (20.00%)
        total disk access time:    6.20 s
        total cryptanalysis time: 22.75 s
        total chain walk step:     18002087
        total false alarm:         17443
        total chain walk step due to false alarm: 12431806

        result
        ---------------------------------------------------
        abyot           ECEMAIL  hex:4543454d41494c
        Administrator   <notfound>  hex:<notfound>
        Guest             hex:
        IUSR_GRIDS                                  <notfound><notfound>
        hex:<notfound><notfound>
        IWAM_GRIDS                                  <notfound><notfound>
        hex:<notfound><notfound>
        TsInternetUser                              <notfound><notfound>
        hex:<notfound><notfound>
        VUSR_GRIDS                                     <notfound>ELMQPPS
        hex:<notfound>454c4d51505053
```

```
__vmware_user__    hex:
```

Machine alpha using Rainbow Table 2

```
lm_alpha#1-7_2_2100x8000000_all.rt:
128000000 bytes read, disk access time: 3.11 s
verifying the file...
searching for 10 hashes...
cryptanalysis time: 21.95 s

statistics
---------------------------------------------------
plaintext found:          2 of 10 (20.00%)
total disk access time:   3.11 s
total cryptanalysis time: 21.95 s
total chain walk step:    18265454
total false alarm:        17435
total chain walk step due to false alarm: 12917599

result
---------------------------------------------------
abyot           <notfound>  hex:<notfound>
Administrator   <notfound>  hex:<notfound>
Guest             hex:
IUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
IWAM_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
TsInternetUser                              <notfound><notfound>
hex:<notfound><notfound>
VUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
__vmware_user__    hex:
```

Machine beta using Rainbow Table 3

```
lm_alpha#1-7_3_2100x8000000_all.rt:
128000000 bytes read, disk access time: 2.75 s
verifying the file...
searching for 10 hashes...
cryptanalysis time: 23.25 s

statistics
---------------------------------------------------
plaintext found:          2 of 10 (20.00%)
total disk access time:   2.75 s
total cryptanalysis time: 23.25 s
total chain walk step:    19364474
total false alarm:        18875
total chain walk step due to false alarm: 13704304

result
---------------------------------------------------
abyot           <notfound>  hex:<notfound>
Administrator   <notfound>  hex:<notfound>
Guest             hex:
IUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
IWAM_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
TsInternetUser                              <notfound><notfound>
hex:<notfound><notfound>
VUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
```

```
__vmware_user__    hex:
```

Machine gama using Rainbow Table 4

```
lm_alpha#1-7_4_2100x8000000_all.rt:
128000000 bytes read, disk access time: 3.28 s
verifying the file...
searching for 10 hashes...
cryptanalysis time: 22.59 s

statistics
---------------------------------------------------
plaintext found:           2 of 10 (20.00%)
total disk access time:    3.28 s
total cryptanalysis time: 22.59 s
total chain walk step:     18751527
total false alarm:         17985
total chain walk step due to false alarm: 13233673

result
---------------------------------------------------
abyot           <notfound>  hex:<notfound>
Administrator   <notfound>  hex:<notfound>
Guest             hex:
IUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
IWAM_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
TsInternetUser                              <notfound><notfound>
hex:<notfound><notfound>
VUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
__vmware_user__    hex:
```

Machine omega using Rainbow Tables 0-4

```
lm_alpha#1-7_0_2100x8000000_all.rt:
128000000 bytes read, disk access time: 0.20 s
verifying the file...
searching for 10 hashes...
cryptanalysis time: 26.31 s

lm_alpha#1-7_1_2100x8000000_all.rt:
128000000 bytes read, disk access time: 3.41 s
verifying the file...
searching for 10 hashes...
plaintext of d90317151676e564 is ECEMAIL
plaintext of 52567951f8073282 is ELMQPPS
cryptanalysis time: 21.72 s

lm_alpha#1-7_2_2100x8000000_all.rt:
128000000 bytes read, disk access time: 4.05 s
verifying the file...
searching for 8 hashes...
cryptanalysis time: 21.31 s

lm_alpha#1-7_3_2100x8000000_all.rt:
128000000 bytes read, disk access time: 3.14 s
verifying the file...
searching for 8 hashes...
cryptanalysis time: 21.13 s

lm_alpha#1-7_4_2100x8000000_all.rt:
128000000 bytes read, disk access time: 4.56 s
```

```
verifying the file...
searching for 8 hashes...
cryptanalysis time: 21.14 s

statistics
-------------------------------------------------------
plaintext found:           2 of 10 (20.00%)
total disk access time:    15.36 s
total cryptanalysis time:  111.61 s
total chain walk step:     92865021
total false alarm:         89613
total chain walk step due to false alarm: 63156611

result
-------------------------------------------------------
abyot           ECEMAIL  hex:4543454d41494c
Administrator   <notfound>  hex:<notfound>
Guest             hex:
IUSR_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
IWAM_GRIDS                                  <notfound><notfound>
hex:<notfound><notfound>
TsInternetUser                              <notfound><notfound>
hex:<notfound><notfound>
VUSR_GRIDS                                     <notfound>ELMQPPS
hex:<notfound>454c4d51505053
__vmware_user__     hex:
```

5.2 Results

The GTK is the de facto standard for deploying and using Grid services. Acting as middleware software, it is going to enable us the sharing, election and aggregation of geographically distributed computing resources depending on their availability, capability, cost, and user requirements for solving large-scale problems/applications. With these claims for GTK, the researcher of this paper went on testing the testbed Grid for the availability of large computational resource and the working functionalities of GTK component packages.

Below is the discussion of results obtained while testing the Grid testbed for the functionalities of GTK components and performance gains while running a large computationally intensive test application – LM Hash cracking.

5.2.1 GTK Components

The testing performed here is for the objective of making sure the components of GTK are functioning (and/or cooperating each other) properly as claimed by their developers. This

testing and verification is very important as the GTK is in its early and continuous development stage.

Globus, being middleware software for a distributed system, is to contain a number of independent (and integrated) packages which by themselves are very complex and difficult to work with. Component functionality testing was performed for GSI, GRAM, GRIS/MDS and GridFTP parts of GTK; and perfect results, which are inline with their specifications and claims, are obtained.

1. GSI Test

   Test Command: `$grid-proxy-init -debug –verify`
   Output: `Successful creation of user proxy certificate`

2. GRAM Test

   Test Command: `$globus-job-run alpha.grid.aau.edu.et /bin/date`
   Output: `Successful display of date`

3. GRIS/MDS Test

   Test Command:
   ```
   $grid-info-search -h alpha.grid.aau.edu.et \
   '(Mds-Os-name=Linux)'
   ```
   Output: `Successful display of information about Grid machines with Linux Operating System only.`

4. GridFTP Test

   Test Command:
   ```
   $globus-url-copy file:///tmp/src/lm_alpha#1-
   7_4_2100x8000000_all \
   gsiftp://alpha.grid.aau.edu.et:2811/tmp/table/lm_
   alpha#1-7_4_2100x8000000_all
   ```
   Output: `Successful copy of LM Hash Table onto the Grid Store.`

5.2.2 Performance

Once the proper functionality of GTK packages is tested, the next test was for the gain of performance from the Grid testbed for the demand of high computational resources – mainly CPU, memory and disk.

Below are performance figures obtained while running the test application, LM Hash Cracking of Windows 2000 user database containing five user passwords, on different computing environments.

1. **Cracking for "lm configuration 0"** (working with only alphabet characters)
   - **a. Results without Grid (using only a single machine)**

Crack Successful	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	111.61	15.36

   - **b. Results with Grid**
     - **i. Involving single Grid machine**

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	118.21	15.86

     - **ii. Involving two Grid machines**

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	67.92	9.53

     - **iii. Involving three Grid machines**

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	45.37	7.81

     - **iv. Involving five Grid machines**

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	22.75	6.20

2. **Cracking for "lm configuration 1"** (working with only alphanumeric characters)

   a. Results without Grid (using only a single machine)

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	153.45	41.51

   b. Results with Grid

      i. Involving single Grid machine

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	149.54	43.80

      ii. Involving two Grid machines

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	103.19	29.14

      iii. Involving three Grid machines

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	78.24	15.84

      iv. Involving five Grid machines

Crack Success	Crypt Analysis Time (sec)	Disk Access Time (sec)
Yes	39.91	9.60

Looking at the above results, applications with high computational resource requirements are to benefit highly from a Grid environment. Reduction in disk access time and execution time is obtained because the load (both execution and reading from disk) are distributed into the machines of the Grid.

From the basic characteristics of parallel/distributed computing, there is no as such perfect scalability. As a result of this, performances obtained on the full Grid testbed are not exact reductions (or divisions) of the values from that of a non-Grid (or single) machine's execution. Causes for non-perfect scalability are complex and so much; but network delays, Grid processing (like wrapping of the actual cracker program by using Globus APIs), and variations of execution environments (each Grid machine has different environment) are the major ones.

Below are summarized plots of the performance results.

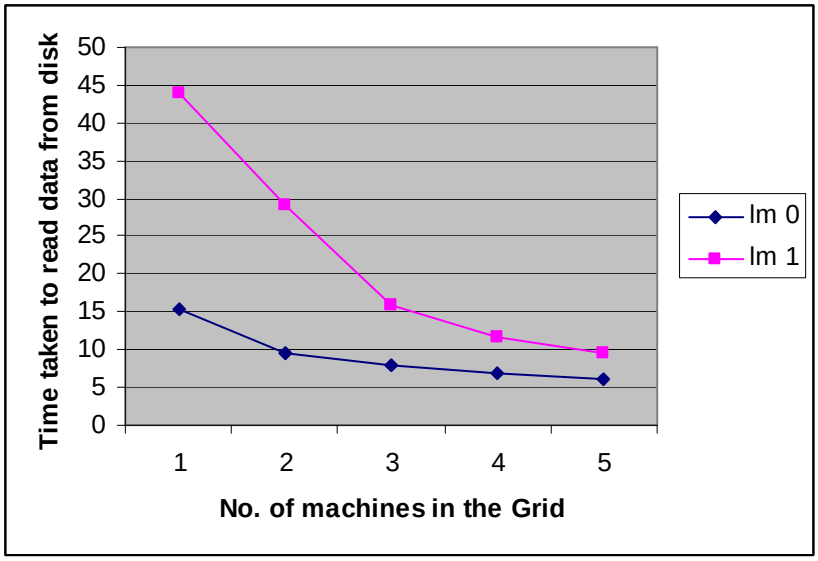

Figure 5.2-1: Performance plot for Disk Access Time

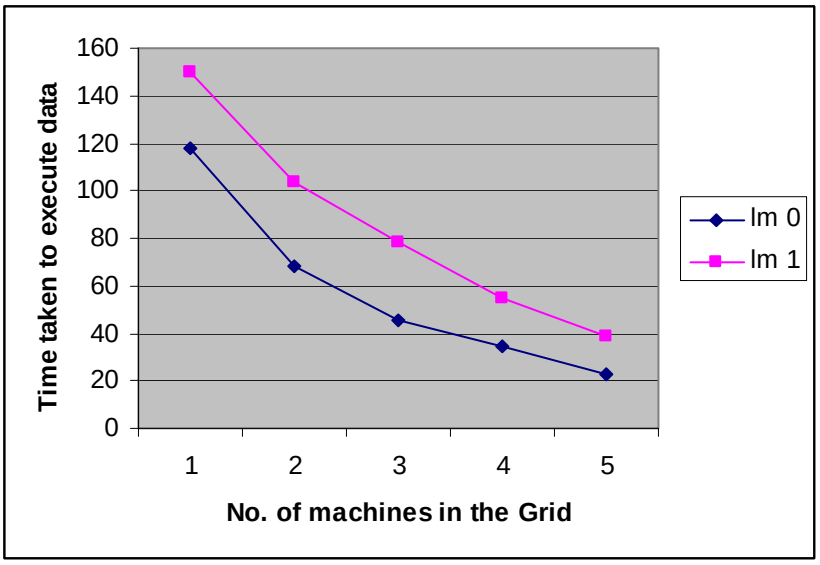

Figure 5.2-2: Performance plot for Cryptanalysis Time

6. CONCLUSION AND RECOMMENDATION

6.1 Conclusion

The major objectives of this thesis work, introducing Grid computing to the users of AAU and studying Grid computing by testing its middleware software Globus Toolkit, are achieved successfully.

Grid computing, in its simplest sense, uses the resources of many separate computers connected by a network to solve large-scale computational problems. With this objective in mind, Grid computing reflects a conceptual framework rather than a physical resource. The Grid approach is utilized to handle a computational task using administratively-distant resources. The focus of Grid technology is therefore associated with the issues and requirements of flexible computational provisioning beyond the local administrative domain mainly through the construction of middleware software called GTK.

The GTK is a fundamental enabling technology for the Grid, letting people share, computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The toolkit includes software components and libraries for resource monitoring, discovery, and management, with security becoming a key and base building block for all these components.

Studying and testing Grid, as adopted in this thesis, is based on the installation and configuration of its middleware software (Grid Testbed Construction) followed by running a sample computationally intensive test application called LM Hash Cracking on the Grid testbed. The installation and configuration of GTK was the very difficult part of this thesis, as it requires a careful study, planning and design works before hand. And all these works are successfully achieved by installing and configuring the GTK on 3 RedHat and 2 SuSE Linux machines of Intel architecture meeting all the requirements of Grid computing.

The testing focuses on the response of Gird for the computational resource requirements of LM Hash Cracking application. And in line with the theoretical specifications of Grid computing, the constructed Grid testbed has responded well enough to the request of

Memory, CPU and Disk. This response is very efficient where a windows password of 10 characters long is successfully cracked in an average time of 10 seconds.

The workings of this thesis, especially introducing Grid to the users of AAU, are limited to the deployment of sample testbed on a few machines of Electrical and Computer Engineering Department and preparation of a well documented design and installation document. Large scale Grid deployment to AAU, possibly with the involvement of stakeholders and decision makers of AAU through debates and forums, are beyond the scope of this thesis. The work that has been done by this thesis, in this regard, is an initial attempt and there remains more for further work and study.

6.2 Recommendation

As an output from this thesis, GTK3 satisfies more or less all the requirements of Grid computing. But implementation of general purpose interface, could be web based (Portal) or shell based, can greatly optimize the performance and functionality of GTK.

Scientists, researchers and programmers of the computing world often develop large models and codes intended to be used by larger communities collaborated and aggregated through Grid computing. Lowering the barrier of entry for access to these codes, modules, or in general resources, is often a technical and sociological challenge and sometimes very difficult to achieve. Portals help bridge the gap because they are well known interfaces enabling access to a large variety of resources, services, applications, and tools for private, public, and commercial entities, while hiding the complexities of the underlying software systems to the user. And hence a general purpose web interface or Portal should be among the basic parts of GTK.

One other point worth mentioning and emphasizing is the security part of GTK. Security in general is very sensitive by its nature and this sensitivity is to be magnified when in the context of Grid computing.

Though security, in Grid, is treated very cautiously using TLS and SSH as major players, there still remains a place which needs further tightening, patching or reengineering. In

Grid the intention is to have “everything-to-everybody” way of computing which is very difficult to achieve in the context of security. And looking at the security principle of GTK, any user running services on GTK servers is first to exist locally (or have a locally mapped account) so that executions can be treated as if under the permission of local user or account. And this approach is going to either put a limitation on the intentions of Grid or create a security loophole if we are going to stick to our intention of availing resources to the public.

This security problem of GTK can be rectified by making use of published accounts or credentials. Meaning, GTK servers (or service providers) are to first have a class of digital certificates, with different privileges, published onto different Certificate servers; some how distributed to avoid single point of servers. So any Grid user which is going to make use of Grid services first needs to go and select accounts according to the credentials available (and/or the need required), and then get or buy (because Grid is to provide/sell service) the appropriate credential and then submit it to the Grid server. With this approach the researcher of this paper strongly believes the current security limitations or loopholes of GTK can be avoided.

As a final say from the researcher of this thesis, Grid computing is to answer huge computational resource requirements which could be very difficult to handle in normal (non Grid) computing environments. This is clearly shown in the results part of LM Hash Cracking performance plot. With this advantage of Grid, the researcher strongly recommends for the construction of campus wide or university wide Grid environment based on the initial efforts presented in Appendix C of this paper.

APPENDICES

A. Testbed Deployment Procedure

Following an incremental deployment approach, the infrastructure server of the Grid is completed first and is followed by the addition of Grid servers. All the servers of the Grid are installed in a similar manner as outlined in Globus server installation section.

Infrastructure server setup

The major steps performed on the Infrastructure server, **alpha.grid.aau.edu.et**, are presented with the following installation steps.

1. Installing RedHat Linux
2. Configuring Network Time Protocol
3. Populating the installation image repository
4. Installing and Configuring Certificate Authority
5. Configuring IDs and Groups
6. Installing Globus prerequisite softwares
7. Installing Globus Toolkit

1. Installing RedHat Linux

The installation of RedHat Linux is straight forward as outlined on many documentations of Linux. It is also important to note that the RedHat is to be installed with **No Firewall** option selected as we are going to depend on many network services.

2. Configuring Network Time Protocol (NTP)

NTP needs to be configured as the clocks on the system within the Grid should be synchronized. The security process of Globus creates proxy certificates that are valid for specific times. So, if the systems do not have their clocks synchronized, users may not be able to use the Grid.

By default, the **Workstation** installation of RedHat Linux is to have NTP package installed. To check the existence of NTP, as root, issue the following command:

```
# rpm –aq| grep ntp
```

An output displaying the version of existing NTP package is to be displayed. If nothing is displayed, it means that NTP doesn't exist and can be installed by downloading the latest version from http://www.ntp.org and by following the installation instruction.

Next, as a root modify the `/etc/ntp.conf` file to properly configure the NTP server. Below are the attributes and their values on the modified ntp.conf file.

```
server          127.127.1.0
driftfile       /etc/ntp/drift
keys            /etc/ntp/keys
```

As root, issue the following commands to start the NTP daemon and change the configuration of the run-time level to start the ntpd daemon automatically:

```
# service ntpd restart
# chkconfig –-level 345 ntpd on
```

3. Populating the installation image repository

A Network File System (NFS) will be used to maintain an installation image repository consisting of all of the software necessary to install GTK3. As root issue the following commands to create the directory structure for NFS

```
# mkdir /GTKP
# cd /GTKP
# mkdir sun-java
# mkdir apache
# mkdir junit
# mkdir globus
# mkdir camgr
```

Next, download the necessary software to be used to the appropriate directory shown in Table A-1.

Table A-1: NFS directory structure for depositing Globus packages

Software	Repository directory	Used for
GPT Source Installation Package	/GTKP/globus	All Globus Servers
Java SDK	/GTKP/sun-java	All Globus Servers
Apache Ant	/GTKP/apache	All Globus Servers
Junit	/GTKP/junit	All Globus Servers
CAMgr Tool	/GTKP/camgr	Infrastructure server

After downloading the required softwares, the next step is to make them accessible from remote Grid machines. The following configuration of NFS, using */etc/exports* configuration file, is used for this purpose.

As root, add the following entry under /etc/exports file

```
/GTKP        *.aau.edu.et(ro,sync)
```

Finally, as root issue the following command so that NFS can be started automatically.

```
# service nfs restart
# chkconfig --level 345 nfs on
```

4. Installing and configuring Certificate Authority

The Globus Toolkit includes the Globus Simple certificate authority (CA) Package. This package provides a simple and useful method for setting up a CA that can be used for test Grid environments. However, in order to install the CA provided in the Globus Toolkit page, at least one bundle of GTK needs to be installed on the machine due to dependency. So, it is decided not to install this CA, but instead the researcher went on choosing OpenSSL, which was provided by Red Hat Linux during the installation. More information on OpenSSL can be found at http://www.openssl.org/.

The following are the necessary macro activities needed to create the CA:

- Create the CA directory structure.
- Copy the CA configuration file.
- Set up the CA.

- Copy the certificate/public key to the /CA directory.

CA directory structure

As root, issue the following commands to create the CA directory structure:

```
# mkdir /CA
# mkdir /CA/IN
# mkdir /CA/OUT
# mkdir /CA/PROCESSED
# chmod 766 /CA/IN
# chmod 744 /CA/OUT
```

CA configuration file

Copy the openSSL /usr/share/ssl/opensl.cnf configuration file to the CA directory. This configuration file contains options for openSSL, in case one wants to customize it.

As root, issue the following command to copy the file:

```
# cp /usr/share/ssl/openssl.cnf /CA
```

Be sure to include the definition of the SSLEAY_CONFIG environment variable in /etc/profile. This variable is used by the internal scripts executed by the openssl command.

As root, add the line into /etc/profile, as shown next:

```
...(author omits lines)

HOSTNAME=`/bin/hostname`
HISTSIZE=1000
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
INPUTRC=/etc/inputrc
fi
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
export SSLEAY_CONFIG="-config /CA/openssl.cnf"

...(author omits lines)
```

Once this line is added to /etc/profile file, it is necessary to source (or refresh) /etc/profile so that the necessary variables are set. For this purpose as root, issue the following command of re-login

```
# . /etc/profile
```

CA setup

The CA script command is used to set up a new CA. This script is provided by OpenSSL and is located in the /usr/share/ssl/misc directory, and it will display a set-up screen where one should enter the information presented in Table A-2.

Table A-2: CA setup prompts

Setup screen prompt	To be entered
CA certificate file name (or Enter to create)	Press the Enter key
Enter PEM pass phrase	<any password phrase>
Verifying - Enter PEM pass phrase	<any password phrase>
Distinguished Name (DN)	Information that will be part of the CA DN. The DN used in this setup is presented in Table 4-4.

As root, issue the following command to setup a new CA:

```
# cd /CA
# /usr/share/ssl/misc/CA -newca
```

This command is to display an output which is summarized by Table A-2.

Public Key

The certificate/public key (cacert.pem) is created during the CA setup and posted in the /CA/demoCA directory. This certificate/public key must be distributed later to all Grid machines, but with a different file name that is formed by the hash number of the certificate plus ".0". The c_hash command can be used to get the hash of the CA.

As root, issue the following command to get the hash number of the CA certificate and copy it to the /CA directory.

```
# cd /CA/demoCA
# HASH=‘/usr/share/ssl/misc/c_hash cacert.pem | cut -c 1-10‘
# echo $HASH
# cp cacert.pem /CA/$HASH
```

Deploy the CAMgr tool

CAMgr is a Java program and set of related classes for managing a simple certificate authority based on OpenSSL. And can be obtained at IBM ftp site at: *ftp://www.redbooks.ibm.com/redbooks/REDP3697*

Installation

This program is intended to be executed by root. Copy the CAMgr.jar and camgr files into the /usr/local/bin directory.

As root, issue the following command to install the CAMgr tool.

```
# cd /usr/local/bin
# cp /GTKP/camgr/CAMgr.jar .
# cp /GTKP/camgr/camgr .
# chmod +x camgr
```

In order to test the CAMgr, as root issue the following command and a screen similar to that of figure A-1 should be obtained.

```
# camgr
```

Figure A-1: CAMgr Graphical User Interface

5. Configuring IDs and Groups

For this purpose issue the following commands as root.

```
# adduser globus
# password globus
# adduser eceGrid
# password eceGrid
```

6. Installing Globus prerequisite softwares

i. Java SDK

The underlying code of GTK is written in Java, so it is necessary to install the Java platform on any machine running the toolkit. For this purpose, the following command is issued from the respective installation image repository folder by root.

```
# rpm –ivh j2sdk-_4_2_09-linux-i586.rpm
```

Next is to add the necessary environment variable for Java. This can be done by modifying the */etc/profile* file as shown below.

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
INPUTRC=/etc/inputrc
fi
JAVA_HOME=/usr/java/j2sdk1.4.1_03
PATH=$JAVA_HOME/bin:$ANT_HOME/bin:$PATH
export JAVA_HOME
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

...(author omits lines)
```

ii. Apache Ant

This is a Java-based build tool that is required for the GTK installation. The following root commands are used in order to install Apache Ant.

```
# cd /usr/local
# tar –xzvf /GTKP/apache/apache-ant-1.6.5-src.tar.gz
# cd apache-ant-1.6.5
# ./build.sh install
```

Following the installation of Apache Ant is addition of environment variable ANT_HOME in /etc/profile.

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
INPUTRC=/etc/inputrc
fi
JAVA_HOME=/usr/java/j2sdk1.4.1_03
ANT_HOME=/usr/local/apache-ant-1.5.3-1
PATH=$JAVA_HOME/bin:$ANT_HOME/bin:$PATH
export JAVA_HOME
export ANT_HOME
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

...(author omits lines)
```

iii. Junit Installation

The installation of Junit is very simple – it only needs unzipping Junit package and copying file junit.jar to Apache Ant library directory. This can be done by issuing the following as root.

```
# cd /usr/local/
# unzip /GTKP/junit/junit3.8.1.zip
# cp junit3.8.1/junit.jar /usr/local/ apache-ant-1.6.5/lib
```

iv. PostgreSQL

Following the installation of Java and its support tools is the installation and configuration of Java Database Connectivity (JDBC) compliant database software. But since the default installations of Linux Operating System are to contain PostgreSQL, the installation steps are skipped here. Further information can be obtained from http://www.postgresql.org/.

7. Installing Globus Toolkit

This section presents the installation and configuration of GTK3. There are dependencies that require the installation to be performed in this order. More information about the installation process can be found in the GTK3 Documentation Web page at: http://www-unix.globus.org/toolkit/documentation.html.

As a security precaution, GT3 needs to be installed by a non-root user. The reason for this lies in the fact that many of the services are designed to be run with limited access to the Globus environment; installing as root negates this important feature.

First, as root, issue the following commands to create the directory:

```
# mkdir -p /usr/local/globus
# chown globus:globus /usr/local/globus
```

Then, as the globus user, issue the following commands to untar the file:

```
$ mkdir /home/globus/install
$ cd /home/globus/install
$ tar -xzvf /GTKP/globus/gt3.0.1-source-installer.tar.gz
```

The toolkit will be installed using the install-gt3 command. As globus user, issue the commands listed below.

```
$ cd /home/globus/install/gt3.0.1-source-installer
$ ./install-gt3 /usr/local/globus
```

Note: The source installation can take several hours depending upon the processor and available memory.

Globus Toolkit post-installation activities

The following are the major post installation services that must be performed after GTK3 installation.

i. Setting environment variables

Environment variables need to be set in order for the location of Globus to be known throughout the system. In addition, the ***globus-user-env.sh*** script needs to run. Below is the modification of ***/etc/profile*** for this purpose.

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
INPUTRC=/etc/inputrc
fi
JAVA_HOME=/usr/java/j2sdk1.4.1_03
ANT_HOME=/usr/local/apache-ant-1.5.3-1
GLOBUS_LOCATION=/usr/local/globus
PATH=$PATH:$JAVA_HOME/bin:$ANT_HOME/bin:$GLOBUS_LOCATION/bin
export JAVA_HOME
export ANT_HOME
export GLOBUS_LOCATION
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
. $GLOBUS_LOCATION/etc/globus-user-env.sh

...(author omits lines)
```

ii. Configuring GSI

Table A-3 below is the summarized GSI setup prompts and values assigned used in the testbed deployment.

Table A-3: GSI setup prompts

Prompt	Enter	Description
Do you wish to continue (y/n)	y	Starts setup
(1) Base DN for user certificates (2) Base DN for hose certificates (q) Save, configure and quit (c) Cancel (h) Help	1	Request for user DN modification
Enter the Base Distinguished Name (DN) for user certificates	ou=ece,o=aau	Modify user DN
(1) Base DN for user certificates (2) Base DN for hose certificates	2	Request for host DN modification

(q) Save, configure and quit (c) Cancel (h) Help		DN modification
Enter the Base Distinguished Name (DN) for host certificates	o=ece	Modify host DN
(1) Base DN for user certificates (2) Base DN for hose certificates (q) Save, configure and quit (c) Cancel (h) Help	q	Save, configure GSI and quit

GSI configuration can be invoked by issuing the following root command.

```
# $GLOBUS_LOCATION/setup/globus/setup-gsi
```

In order to use the certificate authority provided on alpha.grid.aau.edu.et, the CA certificate/public key file, located on the /CA folder of alpha.grid.aau.edu.et need to be copied on to GTK security folder. In order to copy this file issue the following command as root.

```
# scp alpha:/CA/7000141d.0 /etc/Grid-security/certificates/
```

iii. Requesting and Signing Certificates

a. Requesting user Certificate

A certificate for Grid user, eceGrid, is requested by issuing the ***Grid-cert-request*** command. This command is to be issued by the Grid user, eceGrid.

```
$ Grid-ccert-request
```

After issuing the Grid-cert-request command, a file named ***usercert_request.pem*** is stored in the /home/eceGrid/.globus directory.

b. Requesting host Certificate

A host certificate is requested by issuing the ***Grid-cert-request*** command with the ***–service*** and ***-host*** parameters. The host certificate is requested in a similar fashion to the way the user certificate was requested but the commands issued by root as shown below.

```
#Grid-cert-request -service host -host alpha.grid.aau.edu.et
```

c. Signing Certificates

Once the certificate requests are generated, the next step is to send all these requests to CA in order to sign them. The requests can be sent to the certificate signer by using the following root command.

```
# scp /home/eceGrid/.globus/usercert_request.pem \
 alpha:/CA/IN/alphausercert_request.pem
# scp /etc/Grid-security/hostcert_request.pem \
 alpha:/CA/IN/alphahostcert_request.pem
```

Now the CA needs to sign the certificate. This can easily be done by using the CAMgr tool, which can be launched by root using the following command.

```
# cd /usr/local/bin
# camgr
```

iv. Retrieving signed certificates

The signed certificate is stored by the camgr tool in the /CA/OUT directory of the CA machine, alpha.grid.aau.edu.et. The following are commands, by root, to retrieve these certificates.

```
# scp alpha:/CA/OUT/alphausercert_cert.pem \
  /home/eceGrid/.globus/usercert.pem
# scp alpha:/CA/OUT/alkphahostcert_cert.pem \
  /etc/Grid-security/hostcert.pem
```

v. Installing MMJFS

After configuring the certificates, the next step is to install the Master Managed Job Factory Service (MMJFS) by issuing the install-gt3-mmjfs command script. MMJFS is the single point for submitting jobs. It is responsible for exposing the virtual GRAM service to the outside world.

As the globus user, issue the following commands to install the MMJFS source bundle:

```
$ cd ~/install/gt3.0.1-source-installer
$ ./install-gt3-mmjfs /usr/local/globus
```

Upon completion, the script ***setperms.sh*** located in the /usr/local/globus/bin directory need to be executed in order to change the ownership of some Globus files under $GLOBUS_LOCATION/bin directory. This is necessary to allow some Globus tools to run as root. Issuing the following commands as root will serve this purpose.

```
# cd $GLOBUS_LOCATION/bin
# ./setperms
```

vi. Creating a database

In order to create the database, it is necessary to change the login user to postgres and then use the ***creatuser*** command to create a new database user called globus. It is also necessary to give this new database user to give permissions for creating new databases and database users. Below are commands to create this database user.

```
[root@x1 globus]# su - postgres
bash-2.05a$ createuser globus
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER
```

The creation of the database can be completed by issuing the following commands as globus user.

```
$ createdb ogsa
$ psql -d ogsa -f $GLOBUS_LOCATION/etc/rft_schema_ogsa.sql
```

vii. Creating Grid security files

This is the final step of GTK deployment. Two security files need to be created from scratch using any text editor. These files essentially contain a listing of users that are to be given access to the Globus environment and are located in the /etc/Grid-security directory.

Table A-4: Description of Grid Security Files

Security file name	Description
Grid-mapfile	A flat file that maps the Grid users to the distinguished name of the user certificate. Each line represents an authorized user of the resource
grim-port-type.xml	An XML document that maps the user ID to the Grid service name

Any user that is allowed to request services needs to have the information placed in both of these files. For the testbed installation, information about the eceGrid user is included in these files. The following is a single entry of the eceGrid on to ***Grid-mapfile***

```
"/O=aau/OU=ece/CN=ECE Grid" eceGrid
```

Below is the content of file grim-port-type.xml mapping Grid user to Grid service name.

```
<authorized_port_types>
<port_type
username="itso">http://www.globus.org/namespaces/managed_job/
managed_job/
ManagedJobPortType</port_type></authorized_port_types>
```

Note: The entire <port_type username ….> statement must be written on the same line. No blanks or carriage return characters are allowed.

Globus servers setup

All the servers of the Grid, except the Infrastructure server, are to be configured for NTP client so that all the machines in the Grid can be synchronized. Modifying some attributes from file /etc/ntp.conf file, as shown below, can properly configure machines to be an NTP client for the testbed NTP server alpha.grid.aau.edu.et (10.5.14.6)

```
Server          10.5.14.6
driftfile       /etc/ntp/drift
authenticate    yes
keys            /etc/ntp/keys
```

Because all the softwares required for the testbed deployment are collected on the NFS image repository server, alpha.grid.aau.edu.et, the remaining machines of the Grid need to be configured as NFS clients for the NFS server configured so that they can access these collected softwares remotely.

First it is good to create a folder called ***/GTKP*** so that prompts shown above on the Infrastructure server setup can also be applied here. Once this folder is created, the next step is to mount all those softwares stored under the /GTKP folder of the NFS server on to the local /GTKP folder. This can be done by having an entry shown below in the file called ***/etc/fstab.***

```
LABEL=/     /           ext3        defaults                  1 1
LABEL=/boot /boot       ext3        defaults                  1 2
None        /dev/pts    devpts      gid=5,mode=620            0 0
None        /proc       proc        defaults                  0 0
None        /dev/shm    tmpfs       defaults                  0 0
/dev/sda3   swap        swap        defaults                  0 0
/dev/cdrom  mnt/cdrom   udf,iso9660 noauto,owner,kudzu,ro      0 0
/dev/fd0    /mnt/floppy auto        noauto,owner,kudzu        0 0
alpha:/GTKP /GTKP       nfs   rsize=8192,wsize=8192,timeo=14,intr
```

Once the automatic mounting is completed by using the above file, the next step is to start installing the softwares. And these can be done in a similar fashion following installation steps 5 to 7 of the Infrastructure server setup stated above.

B. Globus Job Run Commands

Below are few globus job run commands and their syntaxes.

1. **globus-job-run**

globus-job-run is the basic command for running Globus jobs. globus-job-run runs in the foreground and defaults to sending output to a terminal. It is equivalent to rsh, but has more functionality for running complex jobs on the Grid.

2. **globus-job-submit**

globus-job-submit is for submitting jobs to a remote batch job scheduler such as Condor or the Portable Batch System (PBS) or LSF. With globus-job-submit, one can submit a job, log out, and log back in later to collect the output. That is, globus-job-submit runs in the background and defaults to sending output to the machine running the command. Retrieve output with globus-job-get-output, and then clean up with globus-job-clean.

3. **globusrun**

The globusrun command runs specifications written in the Globus Resource Specification Language (RSL). globusrun can run jobs either in the foreground or background, and can send output to a terminal or to the machine running the command. The trend in Globus software development is toward considering globusrun as middleware, which can be used by application specific shell scripts to manage job submission. In fact, globus-job-run and globus-job-submit are simply shell scripts that use globusrun for job submission, but present a simpler interface to users.

4. **mpirun**

MPICH-G2, an implementation of the Message Passing Interface (MPI) standard based on Globus, has an mpirun command that can be used to submit MPI parallel jobs to multiple computers in a Grid.

Examples

For using globus first we need to have a valid account and X.509 certificate under .globus directory of the account running the globus command.

1. Having a valid account, certificate and proxy and then running the command

   ```
   % globusrun -a -r <hostname>

   globusrun -a -r alpha.grid.aau.edu.et
   ```

   will give "GRAM Authentication test successful message."

2. Now we can try running simple job on the local cluster using globus-job-run command

   ```
   % globus-job-run <hostname> </path/executable> <arguments>

   % globus-job-run alpha.grid.aau.edu.et /bin/echo hello world
   ```

   prints the Output “hello world”

3. Running job on the remote machine beta.grid.aau.edu.et from a local machine

   ```
   % globus-job-run <hostname> </path/executable> <arguments>
   ```

   Example:

   ```
   % globus-job-run beta.grid.aau.edu.et /bin/echo hello world
   ```

   prints the Output: **hello world**

4. To run a script, open an editor and write the script below

   Example:

   ```
   #! /bin/csh -f
   echo "Hello World from "; $GLOBUS_LOCATION/bin/globus-
   hostname
   ```

```
echo arg 1 = $1
echo arg 2 = $2
echo -n "sum is "
echo "$1+$2" | /usr/bin/bc -l
```

and then save the script with **test_script** as the filename. To run the script locally change the permissions on the file with the below command

```
% chmod +x test_script
```

Then use globus-job-run to run the script created above.

For example, if your **test_script** file is located on beta.grid.aau.edu.et

```
% globus-job-run beta.grid.aau.edu.et ./test_script 5 6
```

prints:

Hello World from
beta.grid.aau.edu.et
arg 1 = 5
arg 2 = 6
sum is 11

5. To run a local file on a remote machine there is a concept of staging a file.

Staging Files

If the executable file is on a home machine (the machine from which we are issuing Globus commands) and we wish to run that executable on a remote machine, we will first need to stage our executable. To stage our executable file test_script, which resides on a home machine, over to a remote machine, execute it, and automatically remove the staged copy after the program has finished, we can use the -s (-stage) option that is available with globus-job-run:

Syntax:

```
% globus-job-run <remote host> -stage <executable > <arg arg>
```

For example:

```
% globus-job-run alpha.grid.aau.edu.et -stage ./hw 4 6
```

prints something like:

Hello Globus World from alpha.grid.aau.edu.et
sum is 10

6. Subjobs and Multiple Commands

Now we can run our executable "test_script" in several locations from one command. If our test_script file happens to be on alpha.grid.aau.edu.et:

```
% globus-job-run -args 3 5 \
-: beta.grid.aau.edu.et./test_script \
-: omega.grid.aau.edu.et -np 2 -stage ./test_script 8 9
```

produces output

Hello World from
beta.grid.aau.edu.et
arg 1 = 3
arg 2 = 5
sum is 8
Hello World from
omega.grid.aau.edu.et
arg 1 = 8
arg 2 = 9
sum is 17

Note: - The above command with a delimiter '-:' allows us to run the same command in several locations.

Batch Job Submission using Globus Resource Specification Language (RSL)

The Globus Resource Specification Language (RSL) provides a common language to describe jobs and the resources required to run them. The Globus Resource Allocation and Management (GRAM) uses RSL. There is a flag called -dumprsl which is used to convert command-line arguments into RSL. We can create sample RSL expressions using this option along with the globus-job-run command.

For example:

```
% globus-job-run -dumprsl alpha.grid.aau.edu.et\
/bin/echo "Hello, Globus world."
```

Prints:

```
(executable="/bin/echo") (arguments="Hello, Globus world.")
```

The command tells globus-job-run to print an RSL expression describing the job it would have submitted if the -dumprsl option had not been specified. This RSL script defines the program to be executed (/bin/echo) and the arguments to the program (the string "Hello, Globus world.").

Any globus-job-run command can be converted to RSL in a similar manner. Once an RSL expression is obtained, the globusrun command can be used to submit the job remotely. The globusrun -r option specifies that a resource name follows. The following command shows how to use the RSL expression above.

```
% globusrun -r alpha.grid.aau.edu.et\
'&(executable=/bin/echo) (arguments="Hello, Globus world.")'
```

Checking, Killing, and Cleaning Jobs

1. To check if the job is running and what is the job-id using globus:

If the UNIX prompt has not returned after a globus-job-run or globusrun command, job is still running. Globus can be used remotely to run a UNIX command such as ***ps***, which prints information about active processes. Remote access without login!

```
% globus-job-run <hostname> /bin/sh -c "ps -u <username>"
```

Examples:

```
% globus-job-run alpha.grid.aau.edu.et /bin/sh –c "ps -u eceuser"
```

2. Killing a Job

If we interrupt the globusrun or globus-job-run process by pressing CTRL-C, jobs should automatically be canceled. The job cancellation routines triggered by CTRL-C have a timeout to allow an intelligent job to cleanup. Therefore, some delay must be provided before the globusrun or globus-job-run process exits. Don't interrupt this cleanup by typing CTRL-C again.

C. Grid Planning for AAU

Introduction

Addis Ababa University (AAU) is the oldest higher educational institution in Ethiopia. AAU started its operation in 1950 under the name University College of Addis Ababa. It was renamed Haile Selassie I University in 1962 and then Addis Ababa University in 1975.

AAU runs Diploma, Bachelors, MD, DVM, Masters, Speciality Certificate and PhD degree programs. It launched its first MSc programs in 1979 and its first PhD programs in 1987.

ICT penetration in AAU was generally categorized to be in a slow process. With this regard, the Systems Design and Data Processing Center (SDDPC), that was established in 1970 used to offer batch-oriented data processing services to the central administration in such application areas as, Payroll, General Ledger, etc. The center also provided services to student/staff researchers in analyzing survey results using statistical packages. However, dissatisfaction was indicated by the user community based on a survey conducted to determine the level of effectiveness of the center. [30]

Through time, certain developments have been observed in the widespread penetration of ICT in the various academic and research units of AAU. The use and application of ICT as a delivery vehicle for the curriculum were also emerging. This might be some indicator that led to conclude ICT, specifically networking, was gaining a foothold in AAU. As a result of this popularity and urgency, by 1996 the then administration of AAU setup a committee to study the computerization needs of the university.

By 1997 a Network Study Team setup by AAU computerization team proposed a wide area network connecting all faculties, departments, libraries and administrative units in AAU. And after the processes of design, revision, bidding and procurement the network design gets implemented, through the funding of SIDA/SAREC, and AAU got the Internet connection by 2002. By January 2003 an ICT Development Office of AAU, with the main role of initiating, coordinating and implementing ICT related activities, gets its establishment.

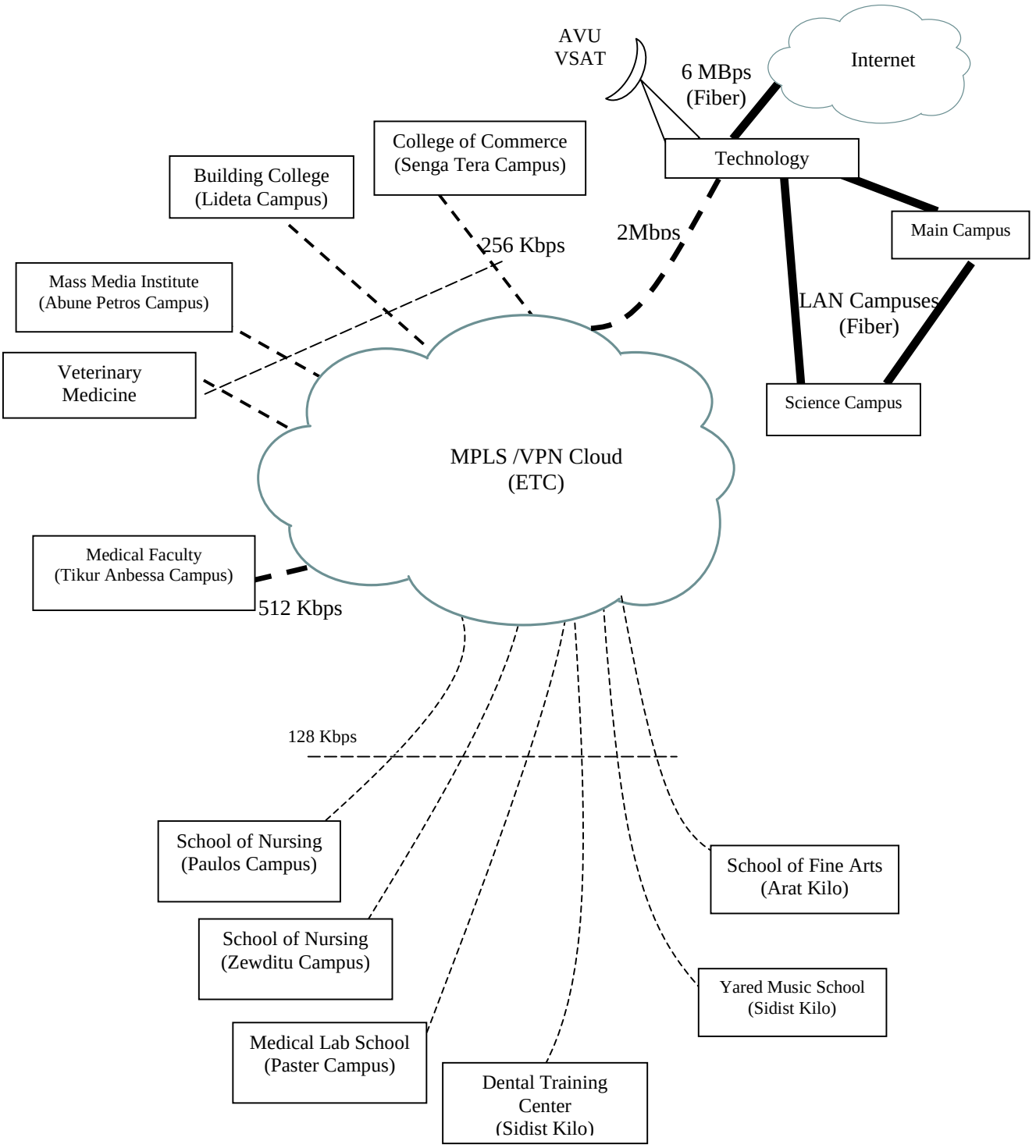

Figure C-1: AAU Campus Network Topology

Despite the fact that faculties of AAU are interconnected with the state-of-the-art networking technology, a lot of work needs to be done by students, researchers and ICT Development Office of AAU for further implementation and utilization of ICT.

As the technology becomes more pervasive and our technological world becomes more complex, without proper and timely acquaintance and understanding of the technology, stakeholders of AAU will find themselves, especially in their careers, at an increasing disadvantage. The work of introducing Grid computing to the users, students and researchers of AAU is one of the timely and economically appropriate works in this regard and hence one of the main objectives of this thesis work.

Preliminary Study

An ad hoc Grid may be installed by a few programmers or individuals in their spare time. But as the Grid becomes large (to cover the dispersed campuses of AAU) and as students, researchers and users become more dependent on it for their mission-critical work, a degree of planning is essential. As a result of this, it is best to understand the AAU stakeholders' computational requirements first and then choose Grid technologies that best fit to these requirements.

To best address the computing needs of users and then come up with the right type of Grid, the researcher of this work has gone through some study facts and analysis steps. Three major attributes identified in this regard are

- current resource utilizations
- current network topology and
- potential problems (computing needs) that can best benefit from the Grid

Resource Utilization

Needless to count and think the amount of investment, AAU currently has a number of computing nodes throughout its campuses; each of them interconnected through the campus wide network. To assess the degree of utilization of these resources, a statistics is

collected from those machines where the computation load is going to be very high; such as servers of AAU, machines of AAU system administrators including the one which the researcher of this thesis used to work on; and the following results are obtained.

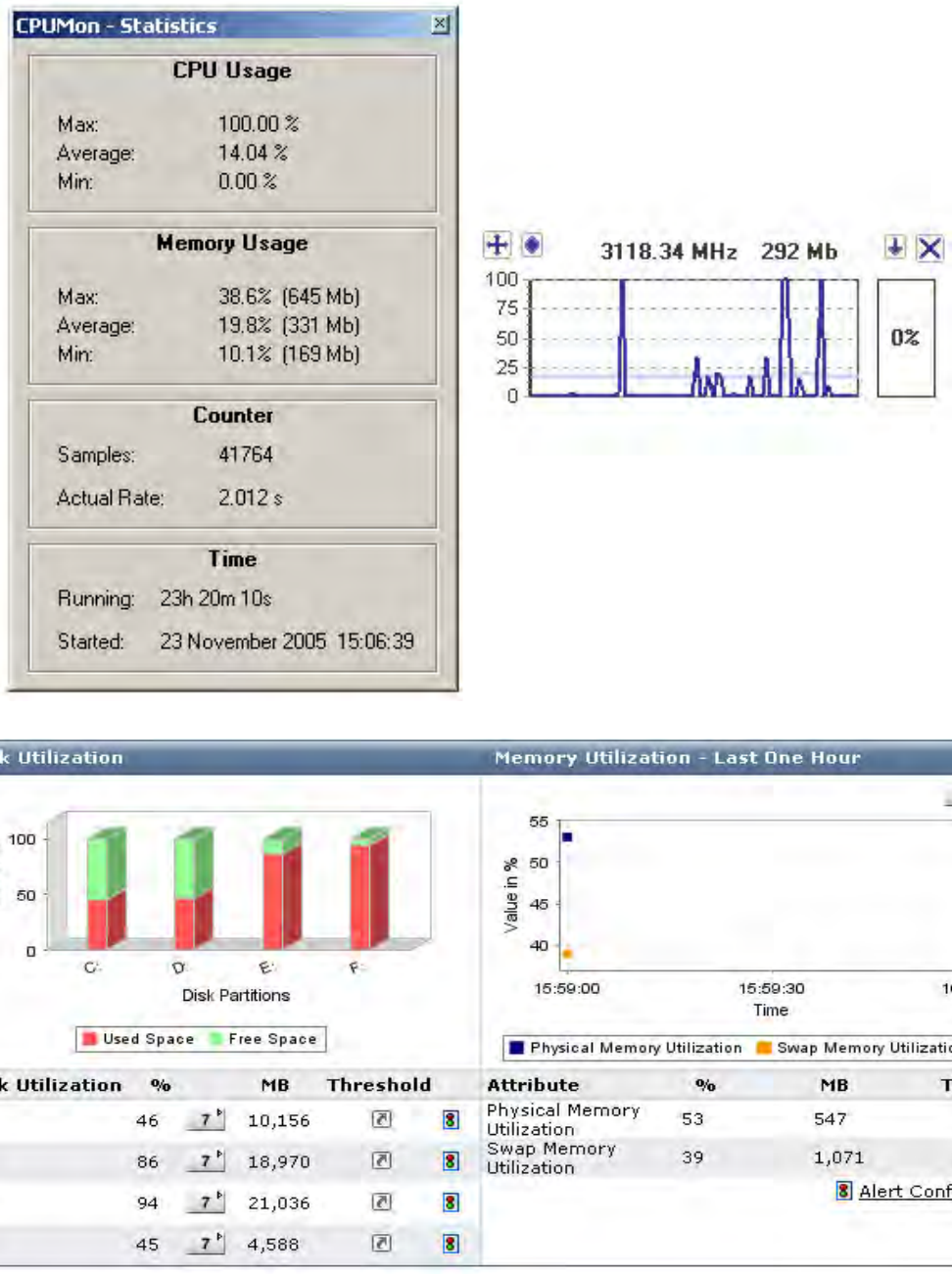

Figure C-2: Resource utilization of a machine used by the researcher of this thesis

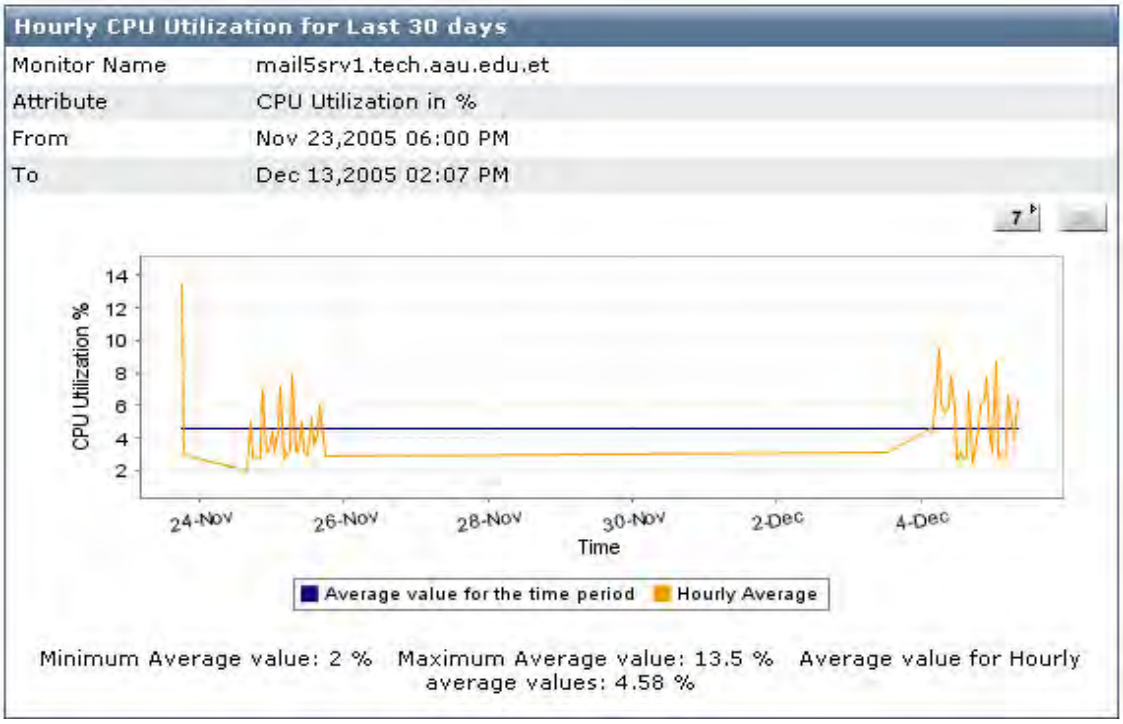

Figure C-3: CPU utilization of one of AAU servers

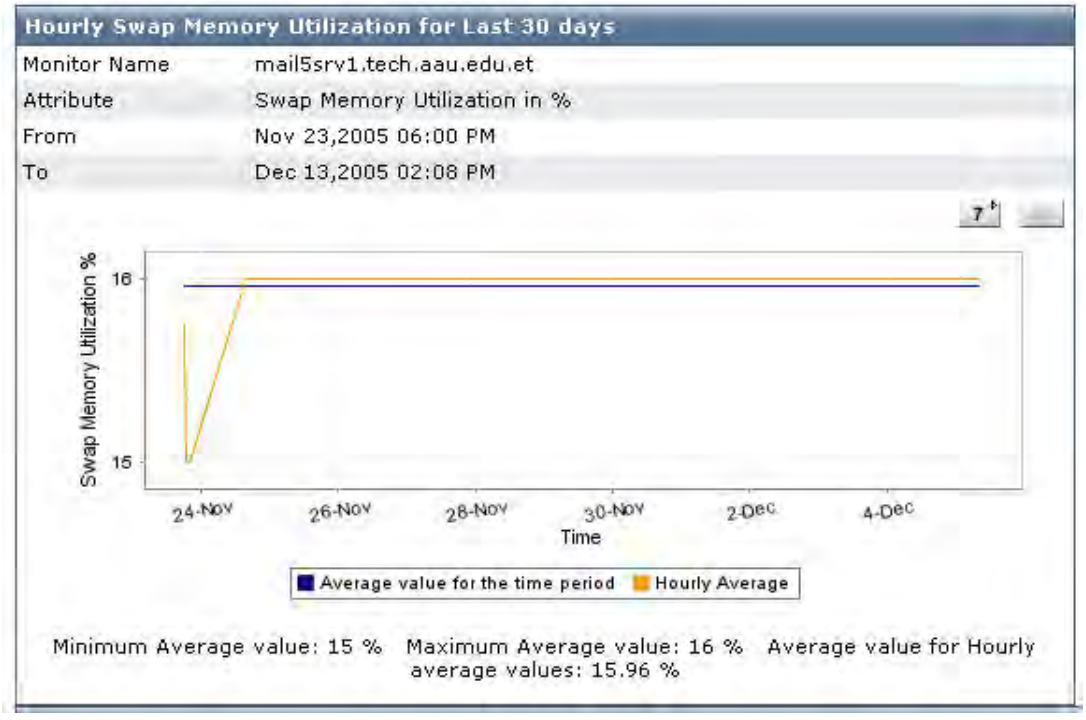

Figure C-4: Memory utilization of one of AAU servers

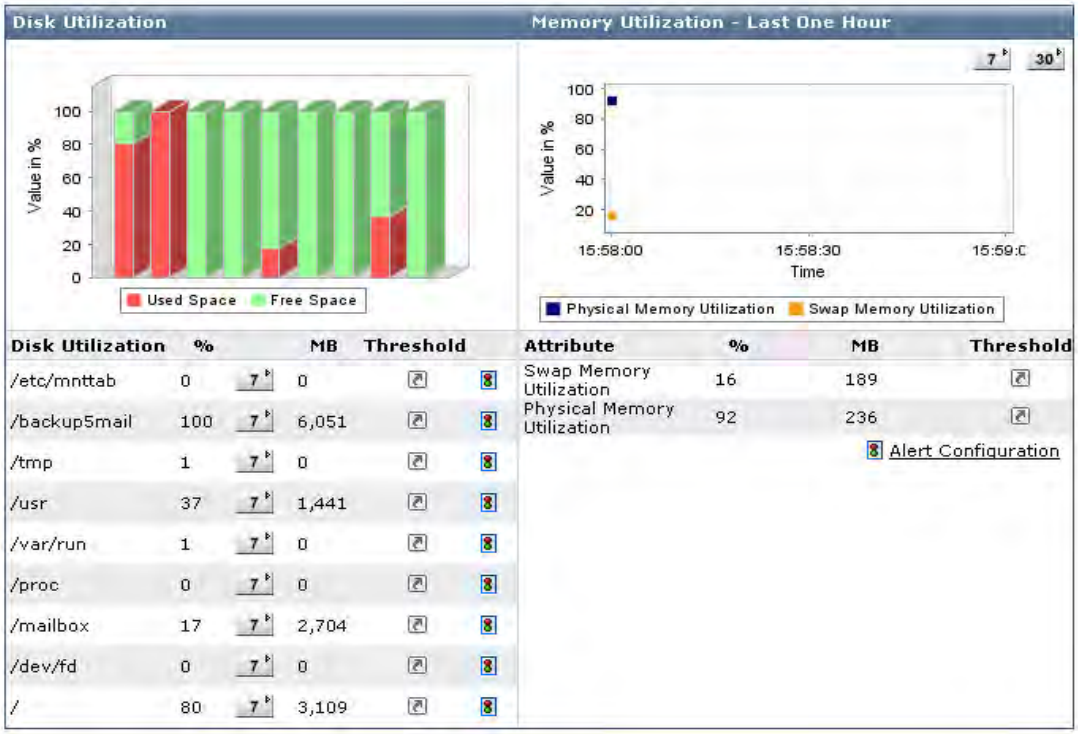

Figure C-5: Disk utilization of one of AAU servers

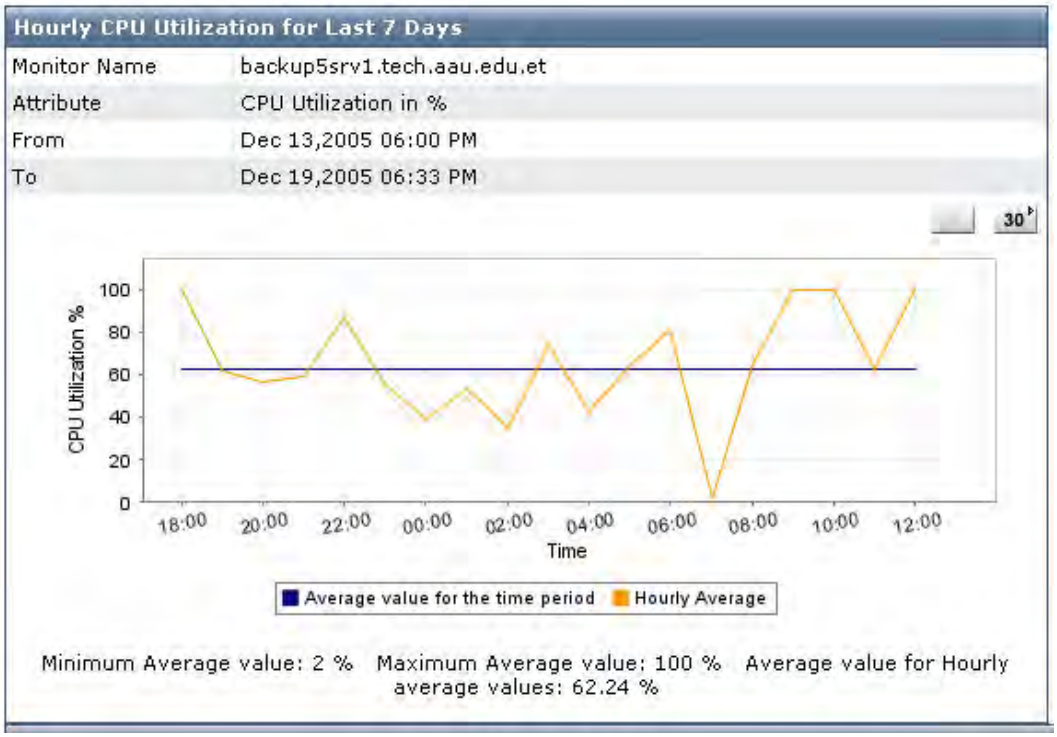

Figure C-6: CPU utilization of AAU's very busy server

As can be seen or figured out from the above statistics, the computing resources have unbalanced and below par utilizations. At the same time, AAU has a number of tasks which are put aside simply because of the lack of large scale, coherent and virtual computing resources, which the use of Grid could have solved at the easiest.

Network Topology

Assuming the three ICT related departments of AAU (namely ECE, CS and SISA) can play a major role in deploying, administering and managing the AAU campus wide Grid, this topology study has focused on the three main campuses of AAU; Technology, Science and Main campus; each one with one of the above ICT departments.

Focusing on the three campuses of AAU doesn't mean that the rest are out of Grid deployment. Instead they can be treated as users (or clients) of the Grid. This is due to the fact that currently these faculties or units do not have that much computing power (relative to the scale of the three campuses identified) to donate.

Below is a possible network topology (specifically Grid topology) that can be deployed on the current existing network architecture of AAU.

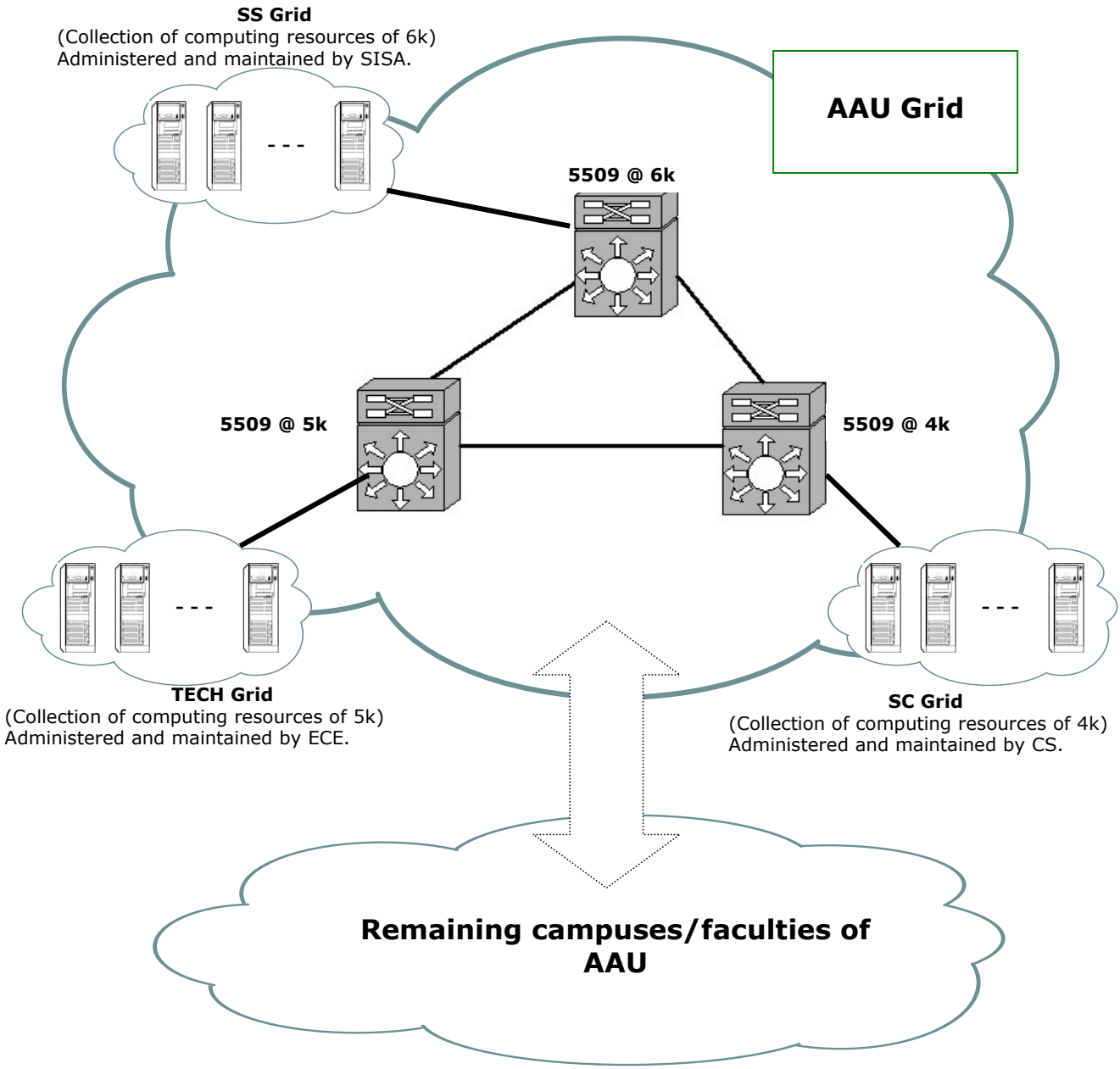

Figure C-7: Design Model for AAU large scale Grid

Tasks for the Grid

Ideally any kind of computing needs can be answered by the Grid. But to manage them somehow efficiently, it is good to identify potential problems (or computing needs) that can best benefit from the Grid. The following are some of the tasks identified in this regard.

a) E-Learning – typical characteristics for this task are high throughput duplex communication, high storage capacity and system integration. In line with this application; integration of existing digital-libraries of technology, computer science and SISA could also create a pool of resources for users of AAU.
b) Scientific simulations – execution of complex system simulations in the areas of signal processing, VLSI design, physics, chemistry and biology.
c) Visualization – tasks from department of Geology, Civil Engineering, Medicine
d) Prediction and Forecasting – task from department of Economics, Statistics, Geology and Geophysics.

Although the Grid is there to give the computing resource (or infrastructure), further work needs to be done (in collaboration with the owner departments and ICT office of AAU) in development, customization and deployment of specific applications for any of the tasks that users of AAU can have.

Deployment Planning

Setting up a computational environment that is physically distributed across the dispersed campuses of AAU and integrating heterogeneous computing resources may cause severe headaches. Apart from the technical issues, many administrative related issues, such as departmental or campus preferences, make it very difficult to accomplish this task. In addition to the technical recommendations suggested above in section 4.1.2, the following deployment steps need to be taken by the ICT Development Office of AAU.

i. Grid Design Workshops
ii. Documentation
iii. Resource Requirement Analysis

a. Hardware requirement – minimal requirement for CPU, RAM, HDD, OS
b. Software requirement – storage, Database, security,
c. Human resource – administrators, developers

After doing a thorough analysis and design in the requirements of software, hardware, interconnection scheme and Grid type identification, it is wise to follow an incremental implementation approach, i.e. it is good to start by deploying a small Grid first, to learn about its installation and management, before having to confront more complicated issues involved with a large Grid. And one final task that needs to be done by the Grid setup team (or body), before starting the actual deployment, is setting up a final prototype. The testbed design and deployment steps presented in Appendix-A can be taken as a role model for this task.

BIBLIOGRAPHY

[1] Andrew S. Tanenbaum / Maarten van Steen, Distributed Systems - Principles and Paradigms, Prentice-Hall of India – 2002.

[2] Andrew S. Tanenbaum, Modern Operating Systems, Second Edition. Prentice Hall, 2001.

[3] Charlie Kaufman/ Radia Pelma/ Mike Spencer, NETWORK SECURITY, PRIVATE Communication in a PUBLIC World, Prentice-Hall of India – 2002

[4] Condor, High Throughput Computing, http://www.cs.wisc.edu/condor/

[5] Fran Berman / Anthony J. G. Hey / Geoffrey C. Fox , Grid Computing - Making the Global Infrastructure a Reality, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England – 2003.

[6] Global Grid Forum, http://www.gridforum.org/

[7] Globus Toolkit Documentation, http://www.globus.org/toolkit/

[8] Grid Café, The place for everybody to learn about Grid, http://gridcafe.web.cern.ch/gridcafe/

[9] Grid Computing Info Centre (GRID Infoware), http://www.gridcomputing.com/

[10] Grid Portal Development Kit (GPDK), http://doesciencegrid.org//projects/GPDK/

[11] GridFTP Documentation, http://www-unix.globus.org/toolkit/docs/3.2/gridftp/

[12] Ian Foster / Carl Kesselman - Computational Grids http://www.globus.org/research/papers/chapter2.pdf

[13] IBM Grid Library, Fundamentals of Grid Computing, http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf

[14] IBM Grid Library, Grid Computing Products and Services,
http://www.redbooks.ibm.com/redbooks/pdfs/sg246650.pdf

[15] IBM Grid Library, Grid Services Programming and Application Enablement,
http://www.redbooks.ibm.com/redbooks/pdfs/sg246100.pdf

[16] IBM Grid Library, Introduction to Grid Computing with Globus,
http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf

[17] Jarek Nabrzyski / Jennifer M. Schopf / Jan WeglarzGrid - Grid Resource Management – Kluwer Academic Publishers.

[18] Kai Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill Series in Computing Science, 1993.

[19] LEGION, A Worldwide Virtual Computer, http://www.cs.virginia.edu/~legion/

[20] LM Hash, http://en.wikipedia.org/wiki/LM_hash

[21] Making a Faster Cryptanalytic Time-Memory Trade-Off,
http://lasecwww.epfl.ch/pub/lasec/doc/Oech03.pdf

[22] ManageEngine™ Applications Manager 6,
http://manageengine.adventnet.com/products/applications_manager/i ndex.html

[23] MyProxy and Grid Portals, http://grid.ncsa.uiuc.edu/myproxy/portals.html

[24] Open Grid Services Infrastructure (OGSI), Version 1.0
http://www-unix.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf

[25] OpenSSL Documentation, http://www.openssl.org

[26] Project RainbowCrack, http://www.antsight.com/zsl/rainbowcrack/

[27] Security for Grid Services,
http://www.globus.org/toolkit/docs/3.0/gsi/GT3_Security_HPDC.pdf

[28] Selecting Secure Passwords,
http://www.microsoft.com/technet/security/smallbusiness/prodtech/WindowsXP/select_sec_passwords.mspx

[29] SETI@home, http://setiathome.ssl.berkeley.edu/

[30] The AAUNet Project, AAU ICT Development Office

[31] The Anatomy of the Grid - Enabling Scalable Virtual Organizations
http://www.globus.org/research/papers/anatomy.pdf

[32] The Globus Resource Specification Language RSL v1.0,
http://www-fp.globus.org/gram/rsl_spec1.html

[33] The Globus Toolkit 3 Programmer's Tutorial,
http://www.casa-sotomayor.net/gt3-tutorial/

[34] The Open Grid Services Architecture, Version 1.0,
http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf

[35] The Physiology of the Grid
http://www.globus.org/alliance/publications/papers/ogsa.pdf

[36] What is the Grid? A Three Point Checklist
http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

DECLARATION

I, the undersigned, hereby declare that this thesis is my original work performed under the supervision of Ato Dawit Mengistu and Dr. Kumudha Raimond, has not been presented as a thesis for a degree program in any other university and all sources of materials used for the thesis are duly acknowledged.

Abyot Asalefew

March, 2006.