Design and Development
of
Resource Repository for Grid Environment

A thesis
submitted in partial fulfillment of the requirements
for the award of degree
of

Master of Engineering
in
Software Engineering

By:
Rohit Sharma
(Roll No. 80631018)

Under the supervision of:
Ms. Inderveer Chana
Senior Lecturer
CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

JULY 2008

Abstract

Grid is a collection of distributed computing resources available over a local or wide area network that appears to an end user or application as one large virtual computing system. Resource management is central to the operation Grid. The first step in resource management is discovery of desired resources. It is the process that takes as input a user request and returns a list of resources or services that can possibly fulfill the given request. Resource Discovery in Grids is critical for efficient resource allocation and management. Grid middleware provides administrators with seamless computing ability and uniform access to resources in the heterogeneous Grid environment. They are software stacks designed to present disparate compute and data resources in a uniform manner. But different middleware have their own mechanism and policy to discover resources in Grid. For example Globus uses MDS, Condor has a match making algorithm to discover the resources whereas Alchemi.NET works on First Come First Serve basis. Problem of heterogeneity further hinders the process of resource discovery.

A good Resource Repository which stores the relevant information about Grid clusters can help very much in resource discovery and further in resource management process. This thesis proposes a Resource Repository which is user friendly and has relevant information of Grid Cluster of Alchemi.NET and Globus, and its centralized structure helps in better management and monitoring of a Grid environment.

Certificate

I hereby certify that the work which is being presented in the thesis titled, **"Design and Development of Resource Repository for Grid Environment"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Name of supervisor* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(**Rohit Sharma**)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(**Ms. Inderveer Chana**)
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(SEEMA BAWA)
Professor & Head
Computer Science & Engineering. Department
Thapar University
Patiala

(R.K.SHARMA)
Dean(Academic Affaris)
Thapar University,
Patiala.

Acknowledgement

I wish to express my deep gratitude to Ms. Inderveer Chana, Senior Lecturer, Computer Science & Engineering Department for providing her immense help, guidance, simulating suggestions and encouragement all the time. She always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under her supervision.

I am grateful to Dr. (Mrs.) Seema Bawa, Professor & Head, Computer Science & Engineering Department, for providing excellent infrastructural facilities and motivation and inspiration that helped me in progressing towards the completion of my thesis work. I am also thankful to Mr. Maninder Singh, Assistant Professor, Computer Science & Engineering Department, a nice person, an excellent teacher and a well-credited researcher, who always encouraged me to keep going with work and always advised me with his invaluable suggestions. I am most indebted to Mr. Prateek Bhatia and Ms. Damandeep Kaur for his help, assistance and suggestions during my work.

I am also thankful to Mr. Balwinder Singh for providing me with the material support. I would also like to express my appreciation to the TUGrid Group. The group held weekly meetings to address various issues and share experiences. Thus, it invoked better understanding of the work and acted as a rostrum for information sharing and problem solving.

I cannot skip mentioning the help and feedback which my colleagues Ms. Shashi, Ms. Anju, Sanmeet, Kunal, Shilpi, Maninder, Neha, Lokesh, Kabir, Vineet and Satyarth have provided throughout my work. I want to thank them all; they were always there at the need of the hour and provided all the help and valuable hints, which I required for the completion of the thesis.

Finally, my special thanks go to my family for their never-ending love and support. Nothing would have ever been possible without my parent's unconditional love and encouragement.

The most valuable thing I acquired from the two years study in TIET is to protect, survive and endure myself in this world with increased confidence and additional faith in my abilities to achieve.

Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Rohit Sharma
(80631008)

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1
Introduction

There has been a dramatic increase in the amount of available computing resources in recent years. Large organizations typically have hundreds or thousands of workstations and global an administrators and desktops use millions of computers. Moreover, according to some research, less than 20 percent of computational capacity is used. If those idle resources could be aggregated and easily used, they would become a huge computational "Grid".

This chapter discusses basics of Grid Computing and its background. It focuses on Grid Architecture and finally discusses the organization of thesis.

1.1 Grid Computing

Grid Computing offers a suite of technologies that explicitly recognizes the new economics of computing and networking and provides the tools that companies can use to drastically cut technology expenditures, increase productivity of technology assets and employees, and have a positive impact on the corporate bottom line [3]. The term grid is used to indicate an analogy with the electrical power grid, linking sources of electrical power together and providing for widespread access [1]. The Grid concepts and technologies are all very new, first expressed by Foster and Kesselman in 1998 [2].

A computational Grid environment behaves like a virtual organization consisting of distributed resources. A **Virtual Organization** is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place [49][51]. Grid is standards based application/resource sharing architecture that makes it possible for heterogeneous systems and applications to share, compute and store resources transparently [1]. The Grid infrastructure promises seamless access to computational and storage resources, and offers the possibility of cheap, ubiquitous Distributed Computing [2]

1.1.1 Background

Since the concept of computational Grids has emerged, the development has been going fast. The term GRID, to denote a distributed computing and storage environment was coined in 1998.The object of a computer Grid is like a power Grid, you plug into a socket and then you have power. Grid was originally conceived and designed in this community to allow access to computing resources that were geographically dispersed. The notion was that underutilized resources in places other than where the researchers were physically located could be used. The Grid is a service for sharing computer power and data storage over the Internet. The aim of Grid is to turn a global network of computers into one vast computational resource and the idea is shown in figure 1.1. Since jobs on the Grid normally involve large-scale resource intensive applications, the need for resource management is ever present [4] [5].

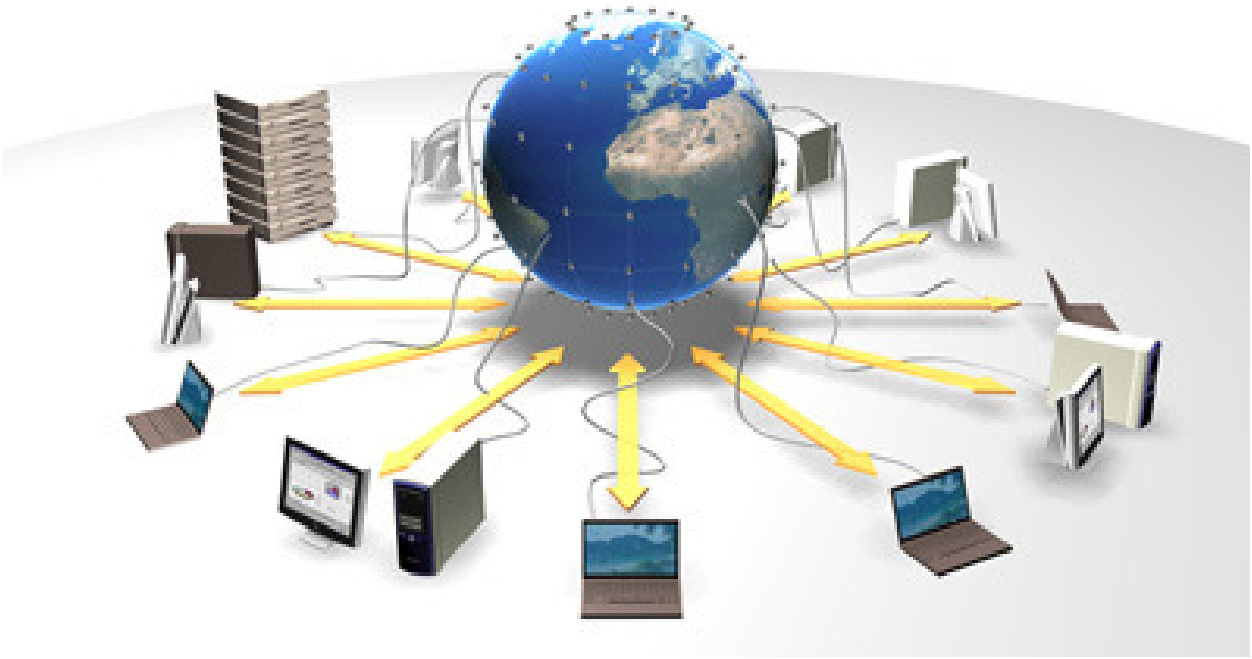

Figure 1.1: Generic view of Grid [5]

The concept behind Grid Computing is not to buy more resources but to borrow the power of the computational resources you need from where it's not being used. Many of

the basic ideas behind the Grid have been around in one form or other throughout the history of computing. One of the "novel" ideas of the Grid is *sharing computing power.* There is a certain amount of "reinventing the wheel" going on in developing the Grid. However, each time the wheel is reinvented; it is reinvented in a much more powerful form, because computer processors, memories and networks improve at an exponential rate [6]. Grid computing has the design goal of solving problems too big for any single supercomputer, whilst retaining the flexibility to work on multiple smaller problems. Thus Grid computing provides a multi-an administrator environment. Its secondary aims are better exploitation of available computing power and catering for the intermittent demands of large computational exercises [7].

1.1.2 Grid Architecture

Grid architecture represents the blueprint by which coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations is made possible in Grid computing [13]. The grid architecture defines the purpose and functions of its components, while indicating how these components interact with one another [12].The architecture of the Grid is often described in terms of "layers", each providing a specific function, as presented in Figure 1.2 as depicted in [33].

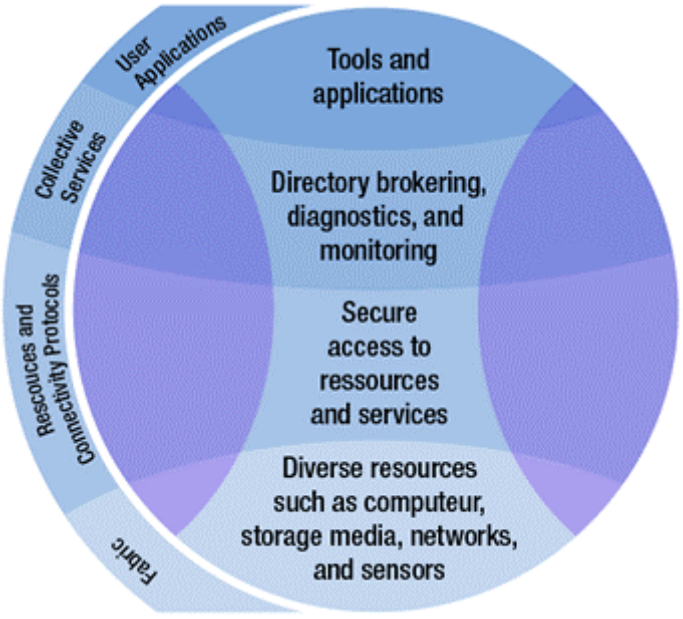

Figure 1.2 Grid Layers [32]

The **Fabric Layer** defines the interface to local resources, which may be shared. This includes computational resources, data storage, networks, catalogs, software modules, and other system resources.

The **Connectivity Layer** defines the basic communication and authentication protocols required for grid-specific networking-service transactions.

The **Resource Layer** contains all the resources that are part of the Grid, such as computers, storage systems, and specialized resources such as sensors [13]. This layer provides the share of single resource based on communication and security service in the connectivity layer [34]. The resource layer calls the fabric layer functions to access and control local resources. This layer only handles individual resources, ignoring global states and atomic actions across the resource collection pool, which are the responsibility of the collective layer.

The **Connectivity Layer** of manufacturing grid realizes the communication between fabric layers and ensures the grid security [34]. While the resource layer manages an individual resource, the collective layer is responsible for all global resource management and interaction with collections of resources. This protocol layer implements a wide variety of sharing behaviors using a small number of resource-layer and connectivity-layer protocols.

Application layer— Application layer provides concrete application service on the basis of *popular* service provided by collective layer [34]. The application layer enables the use of resources in a grid environment through various collaboration and resource access protocols. The Application layer includes all applications that use the resources of the Grid to fulfill their mission. It is also called the Serviceware Layer because it includes all common services that represent mostly application-specific management functions such as billing, time logging, and others [13]. Next section discusses the various characteristics of Grid.

1.1.3 Characteristics of Grids

Different characteristics of computational Grids as have been depicted in figure 1.3

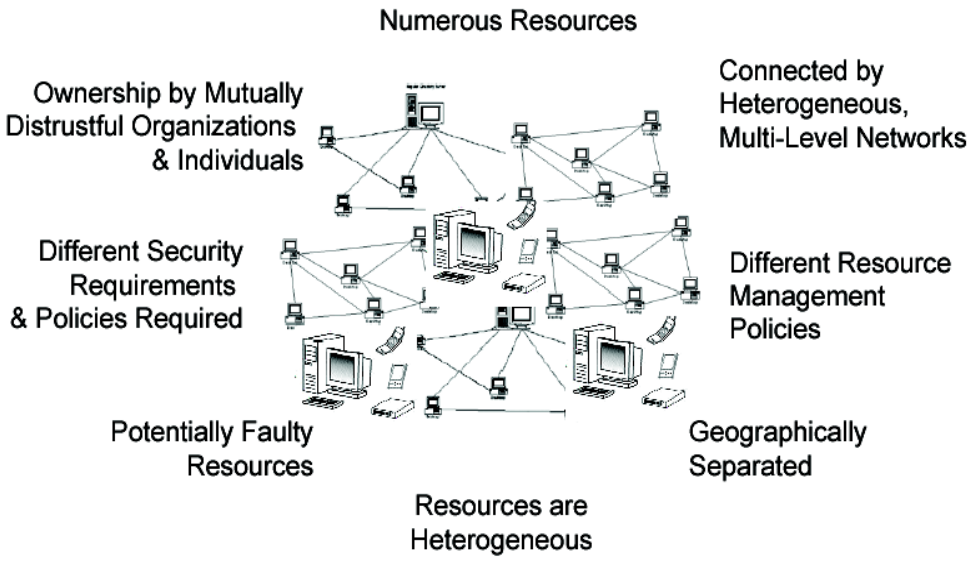

Figure 1.3 Characteristics of Grids [8].

Heterogeneity: A Grid hosts both software and hardware resources that can be extremely diverse ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, super-computers and networks [9]. A Grid involves a variety of resources that are heterogeneous in nature and might span several administrative domains across wide geographical distances.

- Resources are heterogeneous
- Resources are administratively disparate
- Resources are geographically disparate
- An administrators do not have to be concerned about system details (like location, operating system, accounts etc).
- Resources are numerous.
- Resources have different resource management policies.

Resources are owned and managed by different, potentially mutually suspicious organizations and individuals that likely have different security policies and practices [8].

Scalability: It is a desirable property of a system, a network or a process, which indicates its ability to either handle growing amounts of work in a graceful manner, or to be readily enlarged. A Grid might grow from few resources to millions [11]. This raises the problem of potential performance degradation as a Grid's size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant [9] [10].

Dynamicity or Adaptability: In a Grid, a resource failure is the rule, not the exception. As in a Grid there are numerous resources, the probability of some resource failing is naturally very high. The resource managers or applications must modify their behavior dynamically so as to extract the maximum performance from the available resources and services.

Resource coordination: Resources in a Grid must be coordinated in order to provide aggregated computing capabilities.

Reliable Access: A Grid must assure the delivery of services under established Quality of Service (QoS) requirements. The need for reliable service is elementary since an administrators require assurances that they will receive expected, continuous and often high levels of performance [10].

1.2 Motivation

Resources are the base components of a Grid environment. They form the low level entities that are accessed and used to fulfill a an administrator request. Different resources can have the same functional capabilities but they may have different access policies associated, different time access, etc. The key inspiration for grid computing is the sharing of these resources and the first thing needed for efficient sharing is a well-defined resource discovery process. For a well defined resource discovery a very well arranged and an administrator-friendly repository is needed to make information available to an administrators and administrators quickly, clearly and reliably. However, grid resources are potentially very large in number and variety; individual resources are not centrally controlled, and they can enter and leave the grid systems at any time. For these reasons,

resource discovery in large-scale grids can be very challenging. This has been the motivation behind this research work and the focus is on designing and developing resource repository for Grid Environment.

1.3 Organization of Thesis

The chapters in the thesis are organized as follows:

Chapter 2 describes in brief about Resource Management in Grid Computing. This Chapter focuses on Resource Discovery in Grid and its steps, various Resource Discovery Approaches and models used for Resource Discovery are discussed in this chapter. Finally a comparative study of Resource Discovery Approaches in existing Grid middleware has been discussed.

Chapter 3 describes the problem statement and design of proposed Grid Resource Repository for Grid environment, Architecture and its basic components.

Chapter 4 includes the implementation details proposed Resource Repository for the grid environment and the Grid setup.

Chapter 5 summarizes the work presented in this thesis followed by the features that can be incorporated in future for the enhancement of the Grid Resource Repository.

Chapter 2

Literature Survey

This Chapter discusses the Grid Resource Management and various requirements for Grid Resource Management, then Resource Discovery process, steps involved in Resource Discovery process and Resource Discovery Models are discussed. Finally a detailed description of various approaches and Comparison of Resource Discovery approaches in existing Grid Middleware.

2.1 GRID RESOURCE MANAGEMENT

Grid systems are inter connected collections of heterogeneous and geographically distributed resource harnessed together to satisfy various needs of the an administrators. Resource Management is central component of a grid system. It involves managing resources in the system [23]. The term *resource management* is commonly used to describe all aspects of the process of locating various types of capability, arranging for their use, utilizing them, and monitoring their state. More specifically *Grid resource management* can be defined as the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring Grid resources over time in order to run Grid applications as efficiently as possible [22]. Grid applications compete for resources that are very different in nature, including processors, data, scientific instruments, networks, and other services. Complicating this situation is the general lack of data available about the current system and the competing needs of an administrators, resource owners, and administrators of the system [21].

Requirements for Grid Resource Management

Several services or functionalities are required with respect to the Resource Management of Grid systems. The following set is a collection which has been identified in the Grid Scheduling Architecture Research Group (GSA-RG) of the Global Grid Forum [38]

Resource Discovery: It is essential to discover a resource which fulfills specific constraints or fits certain parameters.

Access to Resource Information: In addition to discovering resources, a common requirement is access to available information about a resource. This includes static as well as dynamic information.

Status Monitoring: Prior and during job execution it is necessary to monitor the state of a resource or a job. The resource management framework should support event notification to facilitate reactive measures by the respective services.

Brokering/Scheduling: It is necessary for Grids that a an administrator does not need to manually coordinate the access to resources. Efficient Grid functions are required which automatically select and schedule resource allocation for jobs.

SLA/Reservation Management: While some an administrators might manage resources that handle jobs in a best effort fashion, there are application scenarios in which additional information about the resource allocation is essential. This might include reservations and defined agreements about resource-specific Quality of Service levels.

Execution Management/Provisioning: Clearly the actual execution and management of a job or the requisite provisioning of a resource is the major task of a Grid RMS system. This includes functionalities to cancel awaiting jobs or to claim planned allocations.

Accounting and Billing: In many situations, accountability for the resource utilization is required. Particularly the inclusion of cost considerations for establishing business models for Grids requires functions for financial services. This includes information about the resource consumptions as well as financial management of budgets, payments and refunding. [39]

The overall aim of the resource management is to efficiently schedule applications that need to utilize the available resources in heterogeneous and dynamic environments. Among all components discussed in section, this thesis focuses on Resource Discovery. Next section describes Resource Discovery Process in detail.

2.2 Resource Discovery

The goal of Resource Discovery is to identify a list of authenticated resources that are available for job submission [24]. The discovery of existing resources in a Grid does not necessarily indicate that a resource is actually available or suitable for a specific task or for the requesting an administrator [39]. The actual decision about accessing a resource

will probably require several steps in the resource selection or scheduling process. Therefore, first step is to check whether the required resource is present in the Grid or not. Resource discovery is the process that takes as input a an administrator request and returns a list of resources or services that can possibly fulfill the given request [25]. Next section discusses the steps involved in Resource Discovery.

2.2.1 Steps in Resource Discovery

Most an administrators perform resource discovery in three steps as shown in figure 2.1:

i. Authorization Filtering: It is generally assumed that a an administrator will know which resources he or she has access to in terms of basic services. At the end of this step the an administrator will have a list of machines or resources to which he or she has access.

ii. Application requirement definition: In order to proceed in resource discovery, the an administrator must be able to specify some minimal set of job requirements in order to further filter the set of feasible resources (see Step 3). The set of possible job requirements can be very broad, and vary significantly between jobs. This may include any information about the job that should be specified to make sure that the job can be matched to a set of resources.

iii. Minimal requirement filtering: Given a set of resources to which a an administrator has access and the minimal set of requirements the job has, the third step in the resource discovery phase is to filter out the resources that do not meet the minimal job requirements. The an administrator generally does this step by going through the list of resources and eliminating the ones that do not meet the job requirements as much as they're known. It could also be combined with the gathering of more detailed information about each resource (step 4) (and in fact this is how most proposed systems go about the process). However, when being done by hand, if a an administrator can eliminate an inappropriate resource it is done at this stage to simplify the information gathering in the next phase. [21]

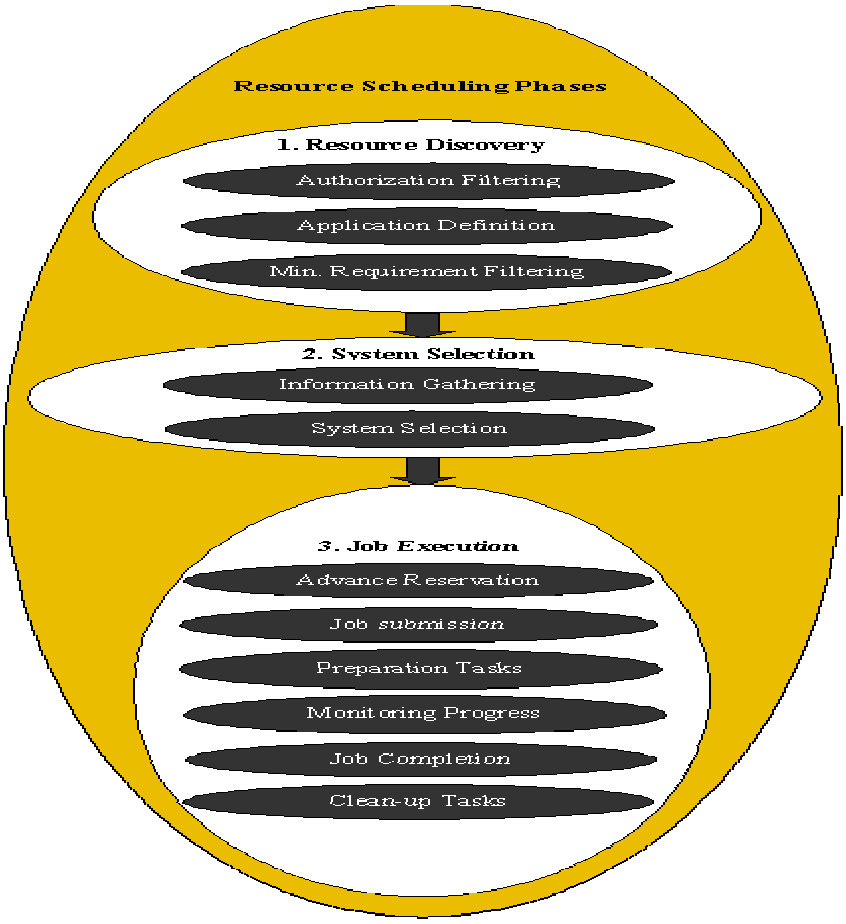

Figure 2.1: Resource Scheduling Phases

2.2.2 Grid Resource Discovery Models

For resource discovery a Grid environment generally uses a pull model, a push model or a push–pull model. The outcome of the resource discovery process is the identity of resources available in a Grid environment for job submission and execution [24].

The pull model

In this model, a single daemon associated with the scheduler can query Grid resources and collect state information such as CPU loads or the available memory. The pull model for gathering resource information incurs relatively small communication overhead, but unless it requests resource information frequently, it tends to give quite out of date information. Figure 2.2 shows the architecture of the model.

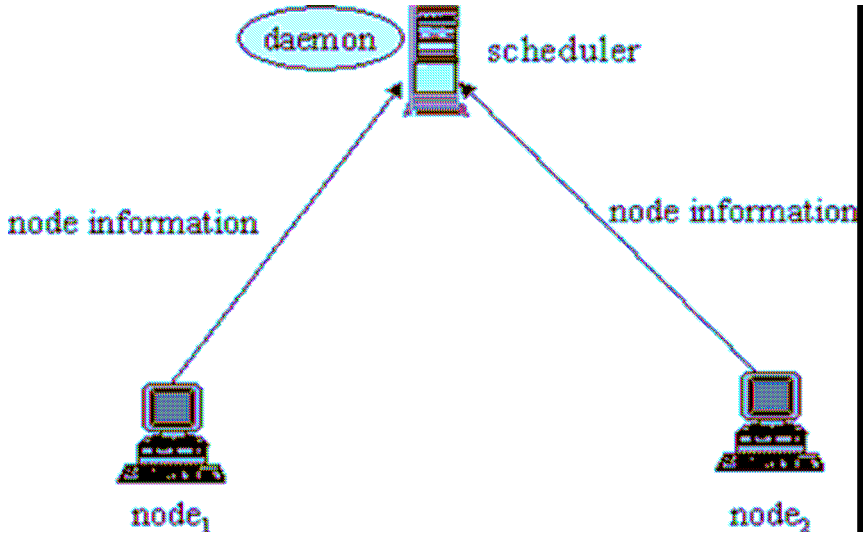

Figure 2.2 The pull model for resource discovery [21]

The push model

In this model, each resource in the environment has a daemon for gathering local state information, which will be sent to a centralized scheduler that maintains a database to record each resource's activity. If the updates are regular, an precise vision of the system state can be maintained over time; obviously, regular updates to the database are intrusive and consume network bandwidth. Figure 2.3 shows the architecture of the push model.

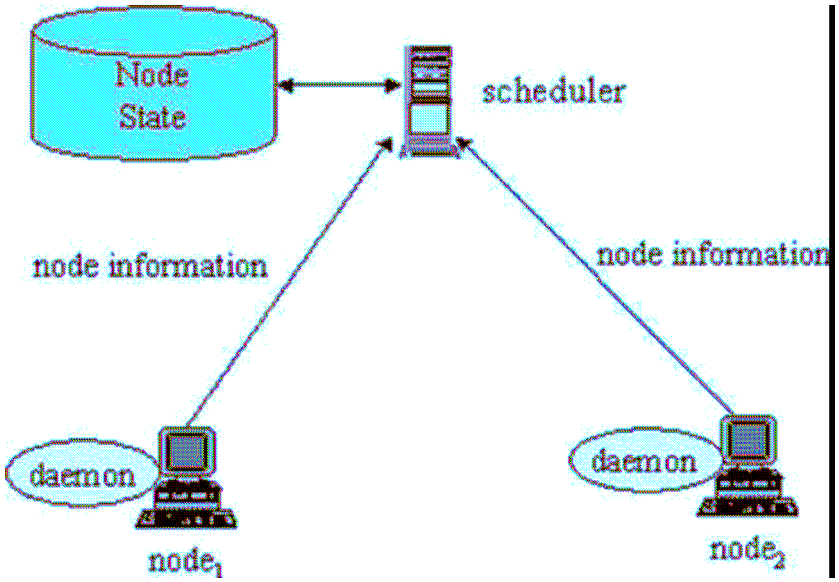

Figure 2.3 The push model for resource discovery [22]

The push–pull model

The push–pull model lies somewhere between the pull model and the push model. Each resource in the environment runs a daemon that collects state information. Instead of directly sending this information to a central scheduler, there exist some intermediate nodes running daemons that aggregate state information from different sub-resources that respond to queries from the scheduler [24]. Figure 2.4 shows the architecture of the push–pull model.

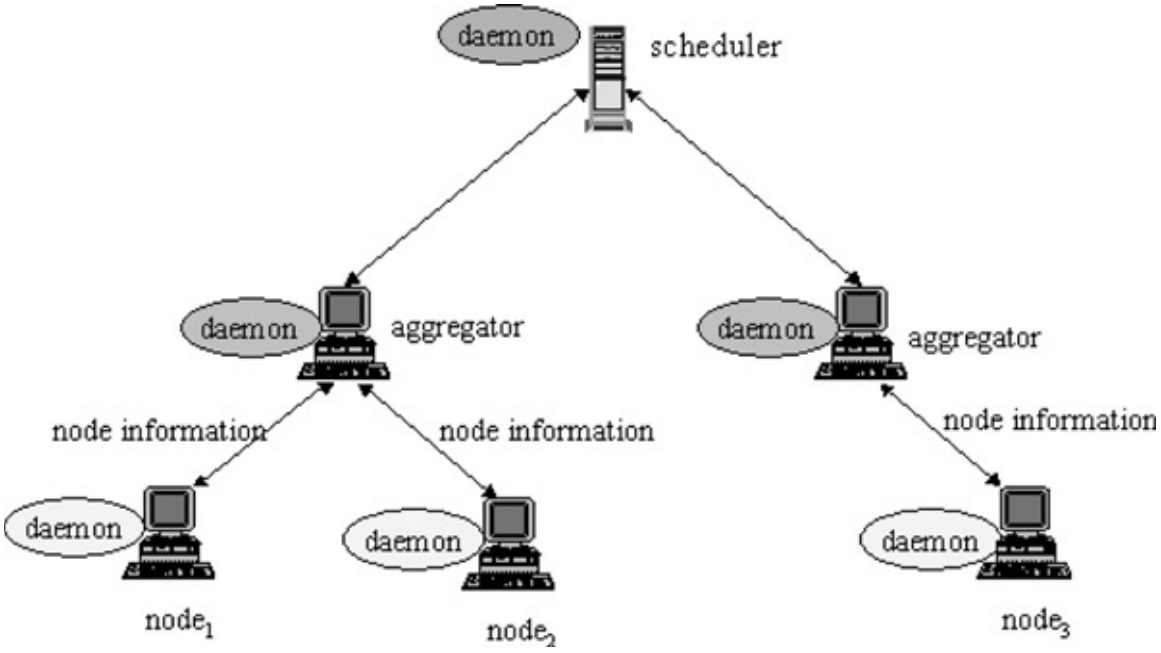

Figure 2.4 The push–pull model for resource discovery [24]

Resource discovery is the first phase of resource scheduling. It involves the an administrator selecting a set of resources. At the beginning of this phase, the potential set of resources is the empty set, and at the end of this phase, the potential set of resources is some set that has passed a minimal feasibility requirement.

2.3 Resource Discovery Approaches

There are various approaches for resource discovery in Grid environments. Some of these approaches are listed as follows:

2.3.1 Query Based and Agent Based Approach

Approaches to resource discovery can be broadly classified as *Query Based* and *Agent Based.* In a *query based* discovery the resource information store is queried for resource availability. Most of the contemporary grid systems use parameterized queries that are sent across the network to the nearest directory, which uses its query engine to execute the query against the database contents. *Query based* system are further characterized depending on whether the query is executed against a distributed database or a centralized database [37]. In *agent based* discovery agents go across the grid system to collect information about resource availability [40] [36].

The major difference between a *query based* approach and an *agent based* approach is that agent based systems allow the agent to control the query process and make resource discovery decisions based on its own internal logic rather than rely on an a fixed function query engine [41,42]. Figure 2.5 shows the Resource Discovery Taxonomy.

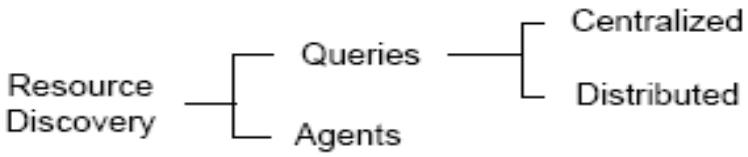

Figure 2.5: Resource Discovery Taxonomy [35]

2.3.2 Peer-to-Peer Approach

Iamnitchi discussed peer-to-peer resource [43, 44] discovery in detail. He proposed peer-to-peer resource discovery architecture for a large collection of resources. Different resource discovery problems are there in large distributed resource-sharing environment especially in Grid environment. Four different architectural components and four environments parameter factors are identified, which dominate the performance and design strategies for a resource discovery solution. Using four axes framework, it is possible to design any resource discovery architecture in a Grid. A general purpose query support enabled “unified Peer-to-Peer Database Framework (UPDF)” for large distributed systems has been proposed.

2.3.3 Parameter-Based Approach

One of the fundamental operations needed to support location-independent computing is resource discovery. Generally, resource discovery schemes maintain and query a resource status database. Dissemination of the resource status information is one of the key operations required to keep the resource status databases consistent [46]. A new concept "Grid potential "is proposed which encapsulates the processing capabilities of different resources in a large network. "Data Dissemination Algorithm" has been proposed in [30] [45]. Grid Potential concept is used to control the extent of data dissemination in Grid systems. Data Dissemination algorithm follows swamping approach for message distribution. When a message comes to a node, that message gets validated. The validation process depends on three types of dissemination; universal awareness, neighborhood awareness, and distinctive awareness. The performance of "universal awareness", "neighborhood awareness", and "distinctive awareness" dissemination schemes was measured.

2.3.4 Quality of Service (QoS)-based Approach

An algorithm to discover the occasionally available resources in a multimedia environment is proposed in [46]. In this paper, different policies for a QoS based resource discovery service for a given graph theoretic Approach [47]. A generalized version of Discovering Intermittently Available Resources (DIAR) algorithm based on occasionally available resources is introduced. Various QoS parameters include processor runtime, storage capacity, network bandwidth and many more. On the basis of these parameters QoS guarantees the best behavior of grid. Through the experiment they found out randomized placement strategies and increased server storage can facilitate better performance to discover a particular resource.

2.3.5 Ontology Description-Based Approach

Ontology refers to a description of a service (resource). The main idea behind this Approach is the advertisement of the resources. In this Approach, service provider registers its service description into the service registry database. When a Grid application sends a request to service directory, matchmaker returns the matches to the

service requester. Requester chooses the best resource based on the specific need. A semantic service discovery framework in a Grid environment is proposed in [52]. Ontology enhances the interoperability between virtual organizations. A service matchmaking mechanism [48] based on ontology knowledge was proposed. It is claimed that this matchmaking framework can provide a better service discovery and also can provide close matches [30].

2.4 Resource discovery in existing Grid middleware

This section discusses about Grid middleware, the resource discovery approaches and policies of different Grid middleware. In this thesis Globus, Condor and Alchemi.NET are discussed.

Grid Middleware

Grid middleware provides an administrators with seamless computing ability and uniform access to resources in the heterogeneous Grid environment. They are software stacks designed to present disparate compute and data resources in a uniform manner, such that these resources can be accessed remotely by client software without needing to know *a priori* the systems' configurations. In Figure 2.6 the middleware module is shown along with the hardware resources [17].

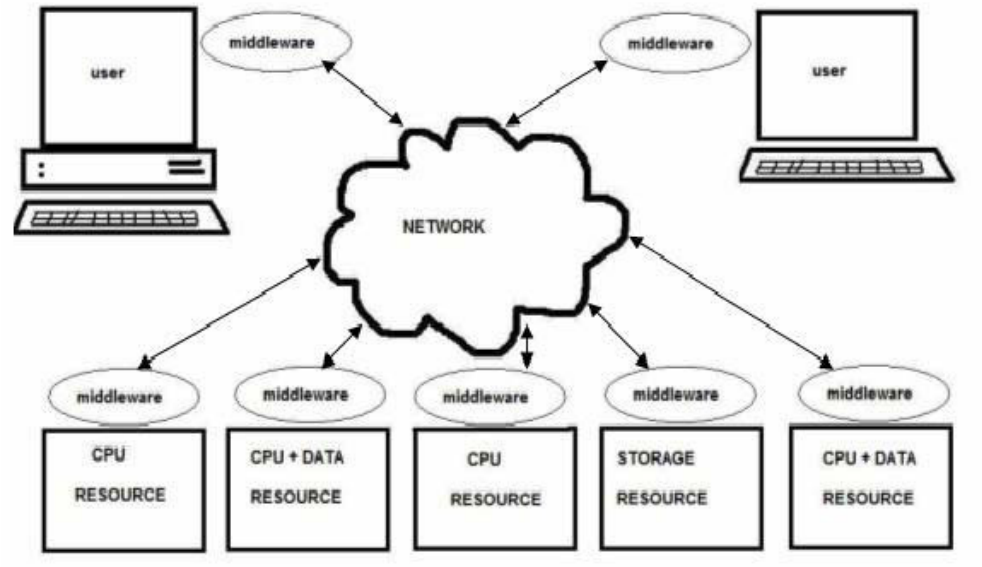

Figure 2.6 A simple Grid environment showing Grid Middleware [17]

2.4.1 Globus

The Globus project is a multi-institutional research to create a basic infrastructure and high-level services for a computational Grid [15].

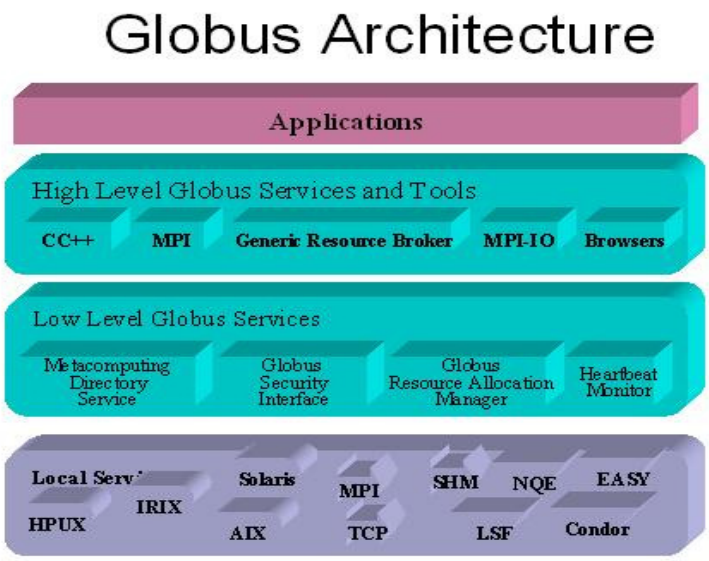

Figure 2.7 The Globus architecture [14]

They have now evolved into an infrastructure for resource sharing among heterogeneous virtual organizations [14]. The architecture of Globus is shown in Fig. 2.7. The Globus toolkit consists of a set of components that implement basic services, such as security, resource location, resource management, data management, resource reservation, and communications.

Resource Discovery in Globus:

Globus uses MDS for Resource Discovery. Globus offers Grid information services via MDS [26]. The Globus Toolkit's Monitoring and Discovery System (MDS) implements a standard Web Services interface to a variety of local monitoring tools and other information sources [27]. MDS currently consists of two components: *Grid Index Information Service* (GIIS) and *Grid Resource Information Service* (GRIS). GRIS provides resources discovery services on a Globus based Grid. The directory information is provided by a Globus component running on a resource or other external information

providers. The resource information providers use a push protocol to update GRIS periodically. GIIS provides a global view of the Grid resources and pulls information from multiple GRIS to combine into a single coherent view of the Grid. Globus is placed into the push resource dissemination category since the resource information is initially periodically pushed from the resource providers. Resource discovery is performed by querying MDS [28].

MDS4 can be considered as a "protocol hourglass" as shown in figure 2.8, which defines standard protocols for information access and delivery and standard schemas for information representation. Below the neck of the hourglass, MDS interfaces to different local information sources, translating their diverse schemas into appropriate XML schema (based on standards such as the GLUE schema whenever possible). Above the neck of the hourglass, various tools and applications can be constructed that take advantage of the uniform Web Services query, subscription, and notification interfaces to those information source that MDS implements [27]. On this base we have a range of Information Providers, used to collect the information from specific resources.

WS MDS consists of:

Index Service - collects monitoring and discovery information from Grid resources, and publishes it in a single location.

Trigger Service - collects data from resources on the grid and, if administrator defined rules match, can perform various actions.

Aggregator Framework - the software framework on which the above WS MDS services are built.

WebMDS - enables end an administrators to view monitoring information via a standard web browser interface, without installing any additional software on their PC.

UsefulRP (4.0.5+ only) - extensible software component that can be used to dynamically generate XML values for one or more WSRF Resource Properties in any given GT4 Java WSRF-Core compatible service.

Information Providers (4.0.5+ only) - used to gather information either via execution aggregator sources (in the Aggregator Framework) or via UsefulRP [28].

The MDS4 hourglass provides a uniform query, subscription and notification interface to a wide variety of information sources, web services, and other monitoring tools.

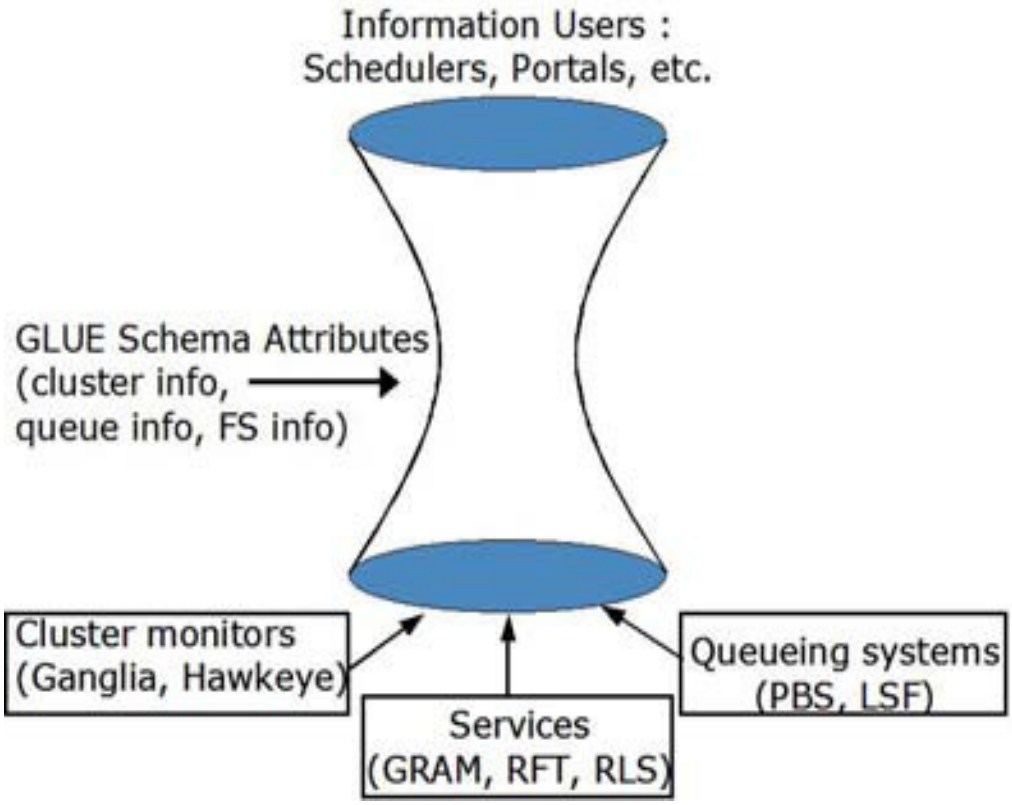

Figure 2.8: The MDS4 hourglass [27].

2.4.2 Condor

Condor focuses on utilization of capacity of unused workstations for computational jobs. It is well suited for jobs which generally do not require communication with each other. Condor provides a job queuing mechanism, scheduling policy, precedence scheme, resource monitoring, and resource management like other full-featured batch systems [17]. Condor [15,16] is a high-throughput computing environment that can manage a large collection of diversely owned machines and networks; Condor is a specialized job and *resource management system* (RMS) [31] for compute intensive jobs. Like other full-featured systems, Condor provides a job management mechanism, scheduling policy, priority scheme, resource monitoring, and resource management [13, 14].

Resource Discovery in Condor:

For Resource Discovery, Condor uses the matchmaking approach as shown in figure 2.9.The basic idea of matchmaking is simple Matchmaking services enable discovery and exchange of goods and services in marketplaces [30].

The main actions involved in the matchmaking process are: A language for specifying the characteristics, constraints and preferences of principals. Requestors and providers advertise their characteristics and requirements in classified advertisements (ClassAds). The classad language is a symmetric description language both servers and customers use the same language to describe their respective characteristics, constraints and preferences. The *Matchmaker Protocol* is composed of the publishing protocol and notification protocol that respectively describe how agents communicate with the matchmaker to post advertisements and receive notifications. The *matchmaker to create matches uses the Matchmaking Algorithm*. In the summary, the matchmaking algorithm relates the contents of submitted classads and the state of the system to the matches that will be created. As part of this process, the algorithm defines a set of conventions (an advertising protocol), which binds meanings to certain classad attributes that will be used for special purposes. The *Claiming Protocol* is activated between the matched parties to confirm the match and establish a working relationship [29].

Condor can have multiple Condor pools and each pool follows a flat RMS organization. The Condor collector, which provides the resource information store, listens for advertisements of resource availability. A Condor resource agent runs on each machine periodically advertising its services to the collector. Customer agents advertise their requests for resources to the collector. The Condor matchmaker queries the collector for resource discovery that it uses to determine compatible resource requests and offers. Compatible agents contact each other directly and if they are satisfied the customer agents initiate computation on the resources [26].

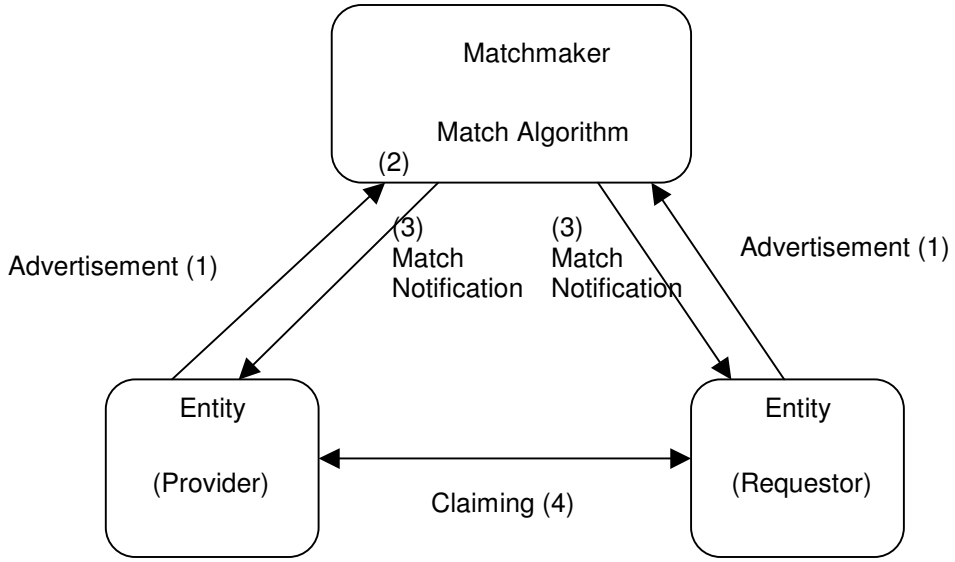

Figure 2.9: Condor Matchmaking [29]

2.4.3 Alchemi.NET

Alchemi.NET is an open source software framework that allows you to painlessly aggregate the computing power of networked machines into a virtual supercomputer (desktop Grid) and to develop applications to run on the Grid. It has been designed with the principal goal of being easy to use without sacrificing power and flexibility [20].

Alchemi.NET is developed on Microsoft .NET Framework and provides the runtime machinery for constructing and managing desktop Grids. It also provides an object-oriented programming model along with web service interfaces that enable its services to be accessed from any programming environment that supports SOAP-XML abstraction [17]. Alchemi.NET was conceived with the aim of making grid construction and development of grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility [32][50]. Four types of nodes (or hosts) take part in enterprise grid construction and application execution:

Manager node: The Manager manages the execution of grid applications and provides services associated with managing thread execution. The Executors register themselves with the Manager which in turn keeps track of their availability.

- *Executor node:* The Executor accepts threads from the Manager and executes them. An Executor can be configured to be dedicated, meaning the resource is centrally managed by the Manager, or non-dedicated, meaning that the resource is managed on a volunteer basis via a screen saver or explicitly by the an administrator.
- *An administrator node*: Grid applications are executed on the An administrator node.
- *Cross-Platform Manager node:* The Cross-Platform Manager is a web services interface that exposes a portion of the functionality of the Manager in order to enable Alchemi to manage the execution of grid jobs (as opposed to grid applications utilizing the Alchemi grid thread model). Jobs submitted to the Cross-Platform Manager are translated into a form that is accepted by the Manager. The Cross-Platform Manager allows custom grid middleware to interoperate with and leverage Alchemi on any platform that supports web services [34].

An *Alchemi Manager* logically couples the Windows Desktop machines running the instance of Alchemi Executor service. An *Executor* service can be configured to receive and execute jobs both in voluntary and non-voluntary modes. Alchemi exposes run-time machinery and a programming environment (API) required for constructing Desktop Grid applications. The core Alchemi middleware relies on the master-worker model - a manager is responsible for coordinating the execution of tasks sent to its executors (desktop machines) [32]. Deploying a Manager node and deploying one or more Executor nodes configured to connect to the Manager construct an Alchemi enterprise grid. One or more An administrators can execute their applications on the cluster by connecting to the Manager. An optional component, the Cross Platform Manager provides a web service interface to custom grid middleware [33] [50].

Resource Discovery in Alchemi.NET:

Alchemi.NET manages its resources on the basis of First-Come-First-Serve (FCFS) basis. For resource discovery is no specific mechanism. It schedules its resources on FCFS basis.

Next section discusses a comparative study of Resource Discovery Approaches in Grid Middleware.

2.4.4 Comparative Study of Resource Discovery Approaches in Grid Middleware

This section proposes taxonomy and comparison of resource discovery approaches in Grid middleware. Taxonomy is primarily concerned with the basic parameters of a resource discovery system.

Taxonomy of Resource Discovery

Platform: Though grid has the concept of heterogeneity, still platform plays an important role. Some times job demands some specific platform. Different middleware are compatible with different operating systems.

Technology: Jobs written in specific language need to be run on specific middleware and use specific technology.

Web Service Interface: Web Services are stated as the latest development in distributed computing on the Internet. Web Services are independent of specific programming languages or operating systems [35].

Resource discovery mechanism: There are various approaches for resource discovery in Grid environments. Like Query Based and Agent Based approach, P2P approach, Ontology Description-Based Approach, Parameter-Based Approach, Quality of Service (QoS)-based Approach and etc.

Security: Care must be taken to secure the stored, processed and transmitted data. Some principles like confidentiality, authentication and integrity must be maintained in all transactions involving secure data.

Extensibility: A Grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant. [31] The Resource Discovery system should be able to scale with growing number of resources, events and an administrators. Therefore

resource discovery system of middleware should be able to handle the large pool of resources.

Comparison of different middleware's resource discovery systems

This section compares Globus, Condor and Alchemi.NET on different qualitative parameters which are mentioned in taxonomy. Table 2.1 provides a comparison between the different middleware's resource discovery systems.

Table 2.1 Comparison of grid middleware's resource discovery systems

Parameters	**Globus**	**Condor**	**Alchemi**
Platform	Unix/ Linux/Windows	Linux/Unix/Windows	Windows with .NET
Technology Used For Implementation	JAVA	C/C++/JAVA	C#, Web Services on Windows + .NET Framework
Web Service Interface	Yes	No	Yes
Interoperability	Yes (via GridBus Broker)	Yes (via Condor-G)	Yes (via GridBus Broker)
Resource Discovery Mechanism	Query Based	Matchmaking	First Come First Serve
Extensibility	High	Good	Medium
Security Level	High	Medium	Medium
Scheduling	Decentralized scheduler Infrastructure	Centralized scheduler	First Come First Serve based scheduling

2.4.5 Resource Repository in Grid Middleware

As discussed in previous section that different middleware have their own mechanism and policy to discover resource availability. A Resource Repository which stores the relevant information about Grid clusters can help very much in resource management

process. A Repository plays a very important part in resource management process. Grid middleware has their own repository to store the valuable information about their cluster, like Globus uses MyProxy which is an online credential repository. Storing Grid credentials in a MyProxy repository allows an administrators to retrieve a credential whenever and wherever they need one, without needing to manage private key and certificate files. Whereas Alchemi.NET uses SQL server to stores information about its cluster. Still there is no common repository in Virtual Organization in multi-middleware scenario.

2.5 Problem Formulation

Globus utilizes the information of Index Service for Resource Discovery. It uses Web MDS to display the job related information as compared to resource related information, but it can not be utilized for deriving performance related inferences. Similarly Alchemi.NET manager shows all the connected resources but it does not give any detailed information of individual resources, which can be utilized for resource management purposes for enhancing performance.

This thesis proposes design and development of Resource Repository for Grid Environment. This repository gives the information of each resource of Virtual organization where resources may be under Globus or Alchemi.NET middleware. Repository stores/retrieve the data of resources of their respective middleware, and that can be accessed on the same system.

Chapter 3

Design of Resource Repository

For efficient resource discovery in multi-grid middleware scenario a common repository needs to be designed and developed as has been discussed. For development of common Resource Repository, Globus and Alchemi.NET middleware have been considered.

- Globus is most widely used to setup a Grid in an Institution, Corporate and virtual organizations. Globus runs on Linux and thus, provides more security and accebility.
- Alchemi.NET runs on Windows platform and has user friendly GUI.

This chapter discusses the design of proposed Resource Repository.

3.1 Design Alternatives

As Globus displays the information of its resources through web MDS. An individual XML file is created for each client connected to server. Further more this file can be fetched either from server or from client itself. By using WSRF query a XML can be generated and this XML file can be parsed by a XML parser program. This XML parser program reads the generated XML file and produces the content of selected tags. Tags can be selected as per requirement/priority.

Alchemi.NET manager has all the information of its resources, which is stored in MS-SQL. Using SQL query, data can be fetched from Alchemi.NET's database and then by update query this fetched data can be stored in designated repository.

There can be two possible design alternatives for solving this problem.

1. Making Database repository on Linux kernel
    a. Fetch information from Alchemi.NET's database and transfer it on Linux.
2. Making Database repository on Windows platform.
    a. Parse XML file, store database on client itself and then transfer this data to server (a Decentralized approach).

b. Get all XML files of clients, on Server, parse them and store data in repository (a Centralized approach).

Any of the above alternatives can be selected for populating and creating the repository. Out of these two alternatives second alternative have been selected for the following reasons:

- This approach is more secure and more reliable.
- Even if Client is down its past/static information can be seen.
- This approach consume less time and space i.e. it saves both time and space.
- No need to execute code again and again.

Chapter 4

Implementation Details and Experimental Results

This chapter describes the implementation details including setting up of the grid environment using various middleware. Design and development of a grid resource repository that resolves the issue of interoperability between various grid middleware as they are based upon different standards and need a common platform as far as the resource discovery is concerned.

4.1 Used Technologies

In this section the summary of technologies used in developing the Grid Resource Repository has been discussed.

Java

Java language offers various features that facilitate with easiness in the development and deployment of a software environment for Grid computing. Java's network based features are very useful to develop Grid application. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture, making Java an attractive basis for portable Grid computing and thus it enables the distribution of computational tasks to different computer platforms [54]. These features are of particular interest for Grid application. As Java has various benefits thus it seems ideal for multi-paradigm communication environments. Many grid and cluster frameworks can host services implemented in Java. In fact, numerous frameworks are implemented entirely in Java. While, the Globus Toolkit 4 (GT4) grid service is implemented in Java, the framework is implemented both in Java and in C.

Reasons why to select Java for implementation are following:

- Java has XML Processing (Parsing, Transforming, and Validating) libraries.
- Java gives very good performance on sequential codes.
- Java provides very high security.
- The Java Database Connectivity (JDBC) API for database access

- The (heavyweight, or native) Abstract Windowing Toolkit (AWT), which provides GUI components, the means for laying out those components and the means for handling events from those components
- Java code can be executed from remote sources securely.
- Java provides a very refined graphical an administrator interface framework.

As security is of paramount importance for the acceptance of a platform for Grid computing, the security and safety features of Java are highly appreciated in this context.

IDE used:

In computing, an ***integrated development environment (IDE)*** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, a compiler and/or interpreter, build automation tools, and (usually) a debugger. Typically an IDE is dedicated to a specific programming language; however some multiple-language IDEs are in use, such as Eclipse, ActiveState Komodo, recent versions of NetBeans, and Microsoft Visual Studio [5]. Here for Java is used for development under NetBeans 6.1 Next section discusses the Netbeans.

NetBeans:

NetBeans is only IDE that runs on Windows, Linux, Mac OS X and Solaris. NetBeans IDE is open-source and free. The NetBeans IDE is a modular, standards-based, integrated development environment (IDE) written in the Java programming language. The NetBeans project consists of an application platform, which can be used as a generic framework to build any kind of application [53].

Next section discusses the MS Access, which has been used to create/manage the data of proposed repository.

MS Access

Microsoft Access is a powerful program to create and manage databases. It has many built in features to construct and view information. Access is much more involved and is a more genuine database application than other programs such as Microsoft Works [5]. Next section discusses the implementation details of development of designed Grid

Resource Repository. This includes steps of installation of Grid middleware to setup a Grid environment, implementation of Proposed Repository.

4.2 Test Bed setup

A Grid environment was setup in the Computer Science and Engineering Department (CSED) and Electronics and Engineering Department (ECED) by installing different middleware on different machines. Details of these middleware's are:

1. Globus (Globus Toolkit 4.0.6) was installed on twelve systems having fedora core-6 operating system within the Centre of Excellence for Grid Computing in CSED, CCCT lab in CSED, Digital Signal Processing (DSP) lab of ECED and software engineering lab in CSED. The step-by-step installation procedure is given in Appendix A.
2. Alchemi.NET 1.06 was installed on thirteen systems within the Centre of Excellence for Grid Computing in CSED, CCCT lab in CSED, DSP lab of ECED and software engineering lab in CSED. Its installation procedure is given in Appendix B.

4.3 Experimental Results

This section discusses the schema of the proposed common Repository for Globus and Alchemi.NET, fields which have been selected for repository. Repository Updater module has also been discussed along with the snapshots.

Table 4.1: Schema for Globus and Alchemi.NET tables

Fields	Description
ipaddress	Stands for Internet Protocol address. This parameter used as primary key in data table.
hostname	A hostname is the unique name by which a network-attached device is known on a network. It is translated form of IP address.

middleware	This field stores the type of middleware to which resource has been connected.
version**	This shows the version of grid middleware which has been installed on resource.
cpu	It represents the CPU power of resource in terms of GHz.
ram	This field represents the RAM capacity of resource in terms of Mb.
OperatingSystem	The operating system on which resource is running.
memory	This field shows the hard disk memory of resource in Gb
an administratortype*	Type of resource over the grid cluster of Alchemi.NET among manager, Executor or An administrator.
reliability	Value of this field shows how reliable the resource i.e. how often it gets down. Its value ranges from 1-5. 1 for lowest and 5 highest reliability.
performance	Value of this field shows how good this resource perform in Grid i.e. accuracy, precision, speed etc. its value also ranges from 1-5. 1 for lowest and 5 for highest performance.
dedicated	This parameter represents whether the resource is dedicated to grid or not.
Groupkey**	Key assign to resource when attached to grid cluster.
GRAMVersion**	Version of Grid Resource Allocation Management.

TotalCPUs**	Total no. of CPUs available on middleware
FreeCPUs**	Number of CPU free at that time
RunningJobs**	Jobs running currently
TotalJobs**	Total numbers of job given to resource
WaitingJobs**	Number of jobs pending
Status**	Enabled or disabled
LRMSType**	Type of local resource management system
LRMSVersion**	Version of local resource management system

* *Alchemi.NET specific field*

** *Globus Specific field*

After creating the database repository first job is to populate this repository with relevant information. To populate and to manage this repository a module has been developed named "Repository Updater". This module can be used for various purposes like addition of new resource, updating the resource information, retrieving and accessing this information etc. It has a very good an administrator friendly GUI.

Before any action an administrator needs to authenticate his/her identity. Very first screen that appears to administrator is Login page on which an administrator is required to fill username and password and to select the action that is to be performed as shown in figure 4.1

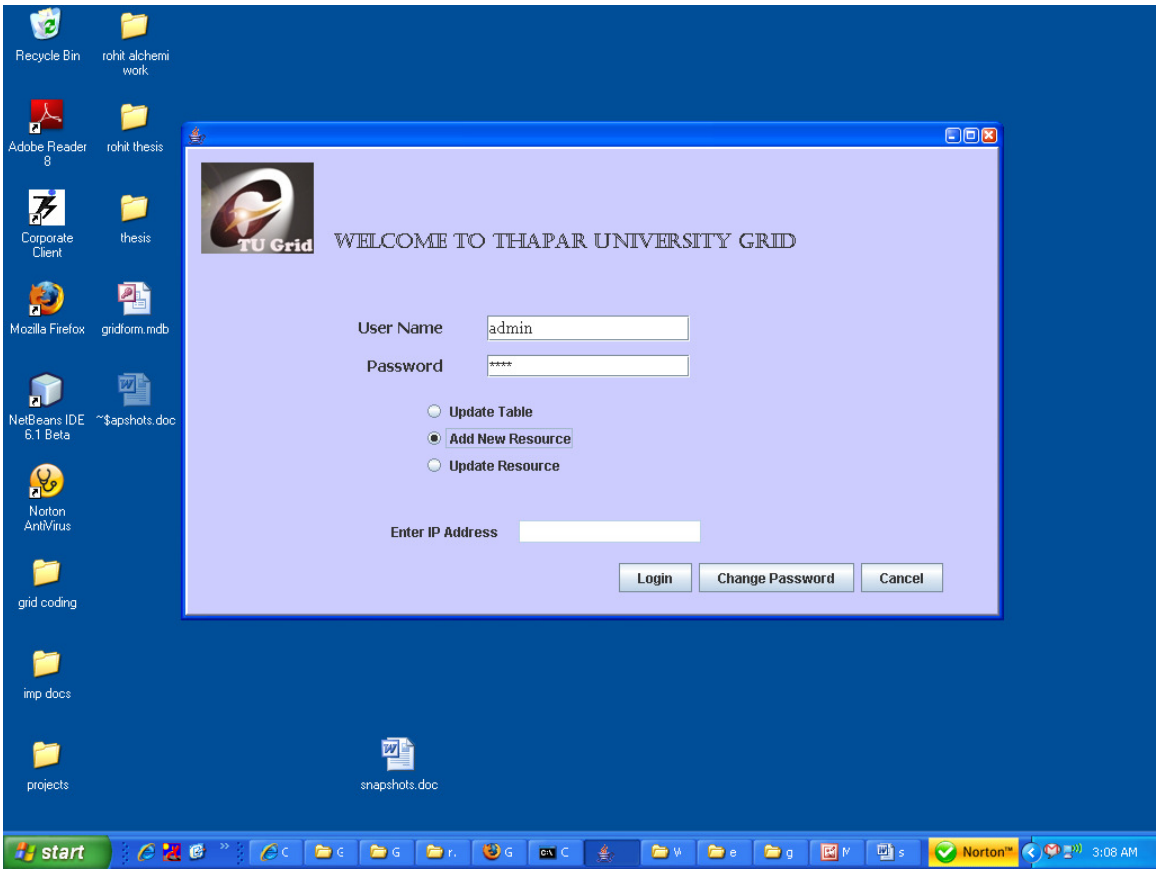

Figure 4.1: login page

As shown in Figure 4.1 there are 3 radio buttons for selection of action and 3 buttons for action an administrator wants to perform, whether he wants to login or cancel or if he wants to change the password only.

If an administrator selects change password button then this module shifts to password_changer page. There an administrator is required to fill old password and enter new password two times, one time for selection of new password, 2nd time for verification. If everything goes right then password will be changed successfully for an administrator. A successful change of password is shown in figure 4.2.

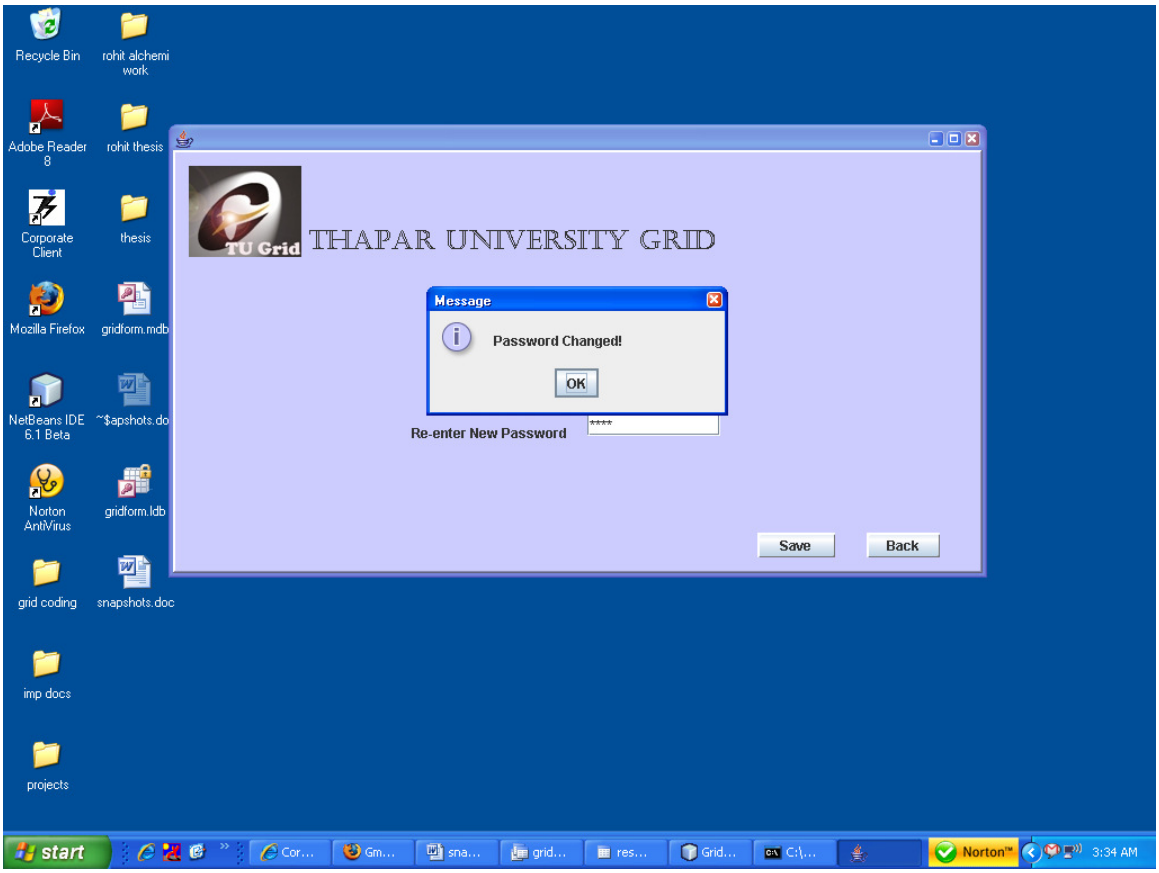

Figure 4.2: password_changer page

If cancel button is selected then module window will be closed. And if an administrator wants to login then he has to select one of the action shown in radio buttons. Without selecting any of them login button will not be enabled. Suppose if an administrator selects the "Add New Resource" and presses Login button then this module automatically shifts to add_resource page, where an administrator needs to enter some basic information about resource, which he wants to add into the Grid Cluster. By pressing "Add" button this resource will be added to repository. A snapshot of this page has been shown in figure 4.3

Figure 4.3: Add_resource page

And if on login page an administrator selects the “Update Table” radio button and presses Login button then this module will shift to Datatable_updater page as shown in figure 4.4. Meanwhile this module calls the XML file paths given for each resource in code, parse those XML files and update the information of already registered resources.

And if, an administrator selects the “Update Resource” option, as administrator selects this option, the text field just below it gets enabled, where an administrator needs to enter IP address of the resource which he wants to update. By pressing Login button, module will search for the IP Address in both Alchemi.NET’s table and Globus’s Table, if it finds the IP Address then it will shift to Add_resource page where value of IP Address text field is already filled with the same value an administrator entered on login page. Now, by clicking refresh button all other fields also gets populated with respect to value of those fields in data tables on this IP Address. Otherwise, if module doesn’t get the IP

Address then module will generate a pop-up message that this IP Address is not registered in repository.

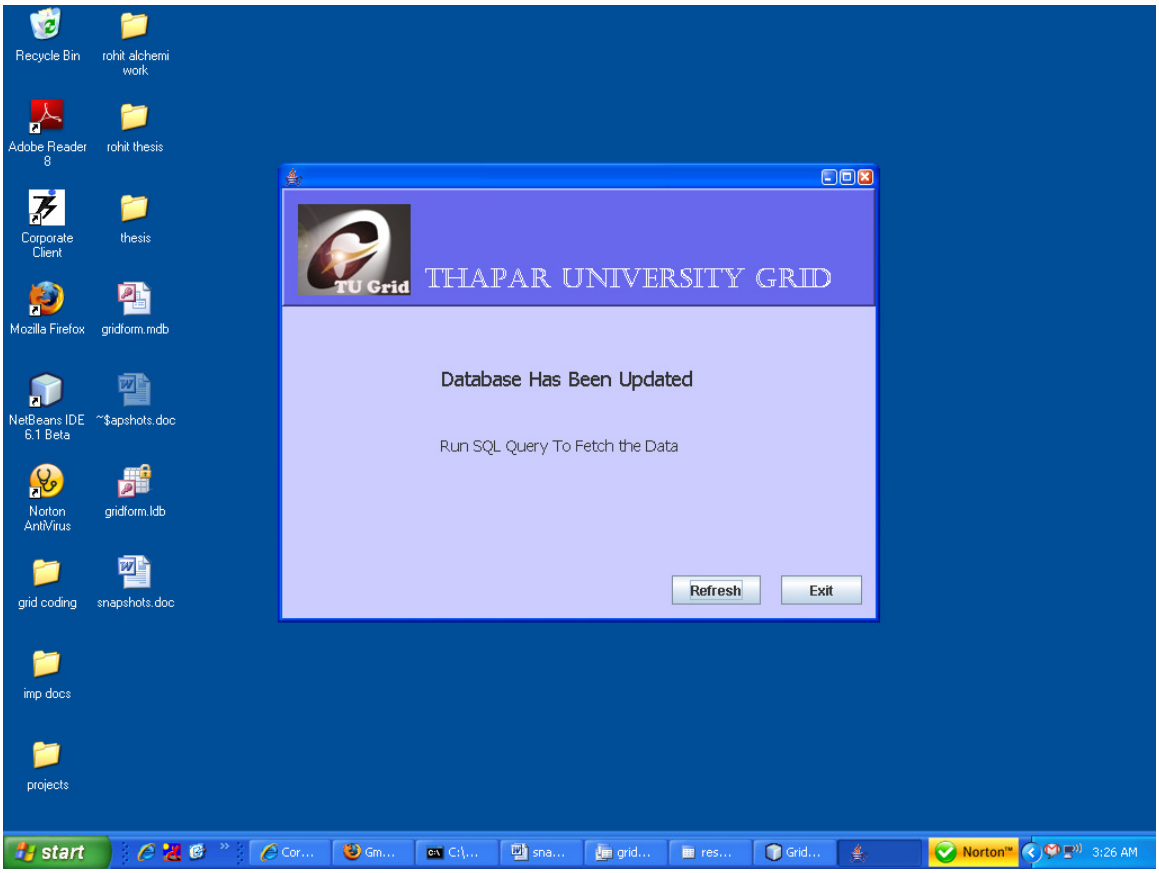

Figure 4.4 Datatable_updater page

Data Base State

Current state of database of Globus cluster and Alchemi.NET cluster can be seen in the figure 4.5 and figure 4.6 respectively.

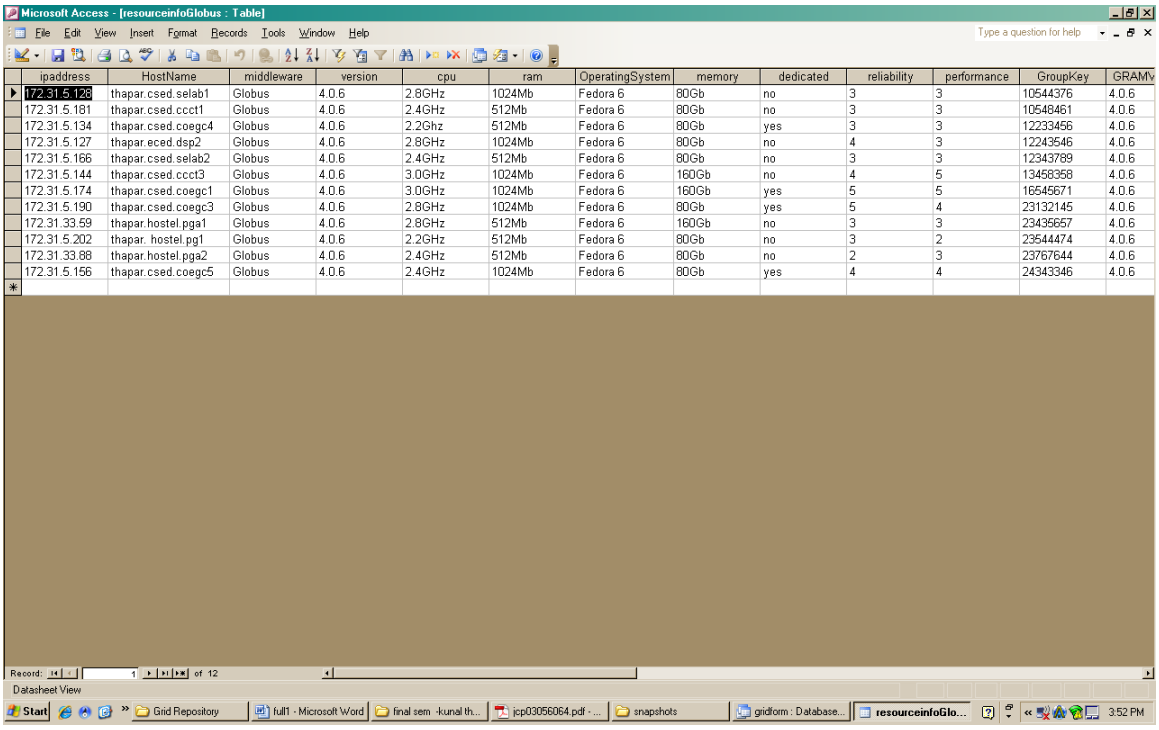

Figure 4.5 Globus Database

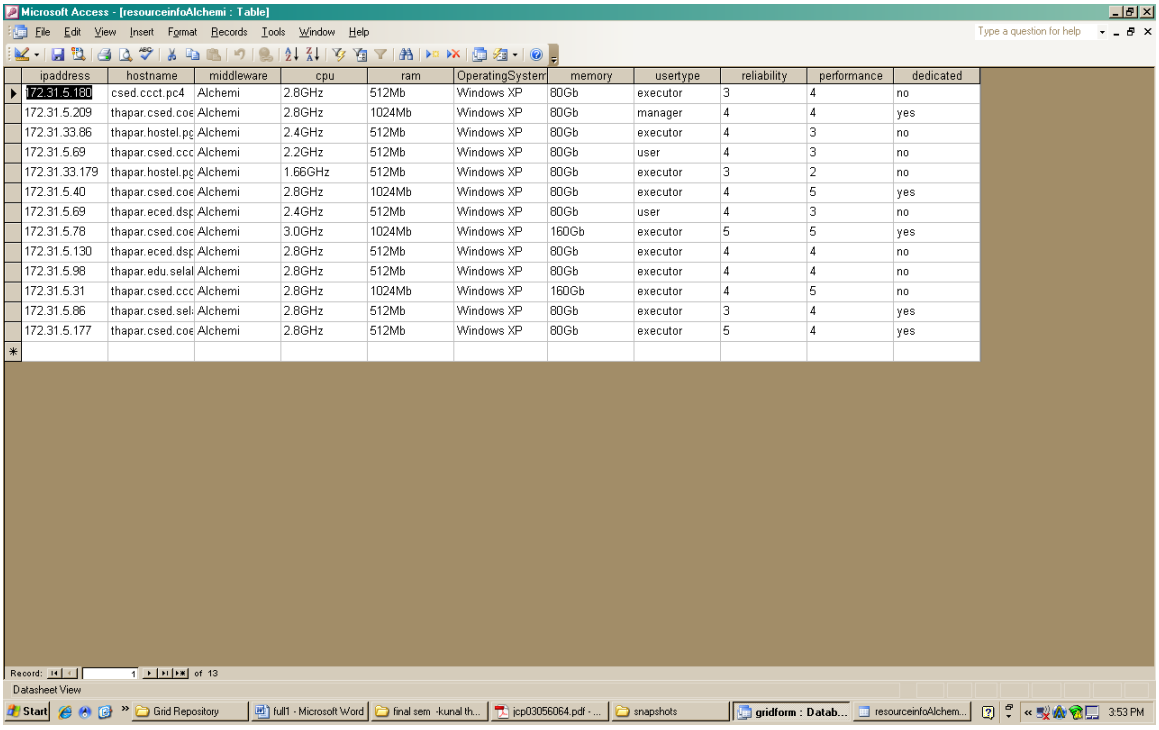

Figure 4.6 Alchemi.NET Database

This chapter has discussed all the implementation details of development of Test Bed and Schema for proposed Resource Repository, where each field of Repository has been discussed in detail and then finally Experimental Results has been shown. Next Chapter concludes the whole thesis and proposes some future work that can be done to enhance the Repository designed.

Chapter 5:
Conclusion & Future Scope of Work

This chapter discusses the conclusions of the work presented in this thesis. The chapter ends with a discussion of the future direction of this work.

5.1 Conclusion

Resource discovery is one of the most important aspects of ongoing Computational Grid research. This thesis work provided an insight into grid computing, the various approaches and models used for resource discovery have been discussed, after that a comparative analysis has been done for different Grid Middleware on the basis of proposed taxonomy. As problem formulation defines that there is problem of common repository for multi-grid middleware scenario then as a solution a common Resource Repository for Globus and Alchemi.NET has been successfully designed and developed. Various design alternatives have been discussed and then the most appropriate method has been selected on the basis of priority, then on the basis of proposed solution a Repository has been developed and the steps involved in implementation has also been discussed. At the end some snapshots were given as experimental results.

5.2 Future Scope of Work

This thesis proposed a centralized repository for two Grid middleware: Alchemi.NET and Globus. As Virtual Organizations are growing day by day and moving towards multi-grid middleware environment so the need for some common repository is also rising. The proposed repository has been designed and developed for two Grid middleware, but it can be further enhanced for multi-grid middleware environment. Apart from it, the repository can be developed for other middleware and not just Globus and Alchemi.NET like Condor, Sun N1 G6 etc.

Reference:

[1]. Lici Lu," Resource Management for a Campus Computational Grid", MS (Computers) thesis, Department of Computer Science, University of Adelaide, June 2002.

[2]. Maozhen Li, Mark Baker, "The Grid -Core Technologies", John Wiley & Sons Ltd, 2005, ISBN-10 0-470-09417-6 (PB).

[3]. IBM and Globus. IBM and Globus announce Open Grid Services for commercial computing, 2002. Available from http://www.ibm.com/news/be/en/2002/02/211.html.

[4]. Peter Birkestrøm Due , Kongens Lyngby, "Trust-based Accounting in Grid Systems", 2006 IMM-MASTER-2006-17.

[5]. www.wikipedia.com

[6]. I. Foster, C. Kesselman, J. Nick, S. Tuecke: "The Anatomy of the Grid: Enabling Scalable Virtual Organization. Accessed from: http://www.mnlab.cs.depaul.edu/seminar/fall2002/GridAnatomy.pdf

[7]. Information about Globus toolkit is Available at www.globus.org

[8]. Global Grid Forum, http://www.ggf.org.

[9]. Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez,"Grid Characteristics and Uses: a Grid Definition" Postproc. of the First European Across Grids Conference (ACG'03), Springer-Verlag LNCS 2970, pp. 291-298, Santiago de Compostela, Spain, Feb. 2004

[10]. *Thilo Kielmann,* Vrije Universiteit, Amsterdam, "Scalability in Grid". PPT CoreGRID, Bridging Global Computing with Grid (BIGG), Sophia Antipolis, France, Nov. 29,2006.

[11]. Lichen Zhang," Scheduling Algorithm for Real-Time Applications in Grid Environment", 2002 IEEE SMC TP1K5.

[12]. J. Joseph, M. Ernest, C. Fellenstein, "Evolution of grid computing architecture and grid adoption models" IBM SYSTEMS JOURNAL, VOL 43, NO 4, 2004,

[13]. http://www.sei.cmu.edu/isis/guide/engineering/architectures.htm

[14]. http://gridcafe.web.cern.ch/gridcafe/gridatwork/images/gridfabric.gif

[15]. Ian Foster, Carl Kesselman , Steven Tuecke," The Anatomy of the Grid", http://www.globus.org/alliance/publications/papers/anatomy.pdf.

[16]. Xiao-lin Zheng, De-ren Chen, Lin Lu," Research of Architecture for Grid Manufacturing", the 9th international on Computer Supported Cooperative work in Design, vol. no 1, p.p. 345-349, May 24-26, 2005.

[17]. Vikas Agrawal, "Establishment of QoS enabled multimedia collaboration Grid over native IPv6 fabric", Thesis submitted at Birla Institute of Technology and Science Pilani, Rajasthan, India

[18]. Parvin Asadzadeh, Rajkumar Buyya, Chun Ling Kei, Deepa Nayar, and Srikumar Venugopal, "Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies, High Performance Computing: Paradigm and Infrastructure", ISBN: 0-471-65471-X, Wiley Press, New Jersey, USA, June 2005.

[19]. http://www.npaci.edu/all-hands/FY98/results/globus/img005.JPG

[20]. Information Alchemi.NET available at http://www.alchemi.net

[21]. Jarek Nabrzyski, Jennifer M. Schopf, Jan We¸Glarz, "Grid Resource Management State of the Art and Future Trends", Kluwer Academic Publishers Boston/Dordrecht/London, ISBN: 1-4020-7575-8, 2003.

[22]. M. B. Juric, I. Rozman, M. Hericko, T. Domajnko, "C*ORBA, RMI and RMI-IIOP Performance Analysis and Optimization*", University of Maribor, Slovenia. 2001

[23]. Chaitanya Kandagatla, "Survey and Taxonomy of Grid Resource Management Systems", http://www.cs.utexas.edu/~browne/cs395f2003/projects/KandagatlaReport.pdf.

[24]. Ian Foster and Carl Kesselman, The Grid:Blueprint for a New Computing Infrastructure", Morgan Kaufmann, San Farnsisco, CA, ISBN-13:978-1558604759, 1999.

[25]. Ioan Toma, Kashif Iqbal, Matthew Moran and Dumitru Roman, "An Evaluation of Discovery approaches in Grid and Web services

Environments", Digital Enterprise Research Institute (DERI), Galway, Ireland and Innsbruck, Austria., http://www.uibk.ac.at/~c703305/publications/eval-gsem2005.pdf

[26]. Klaus Krauter1, Rajkumar Buyya and Muthucumaru Maheswaran1, "A taxonomy and survey of grid resource management systems for distributed computing", Software Practice and Experience, Vol. 32, No. 2, Feb. 2002, pp. 135-164.

[27]. MDS4: The GT4 Monitoring and Discovery System http://www.globus.org/toolkit/docs/4.0/info

[28]. Information Services, http://www.globus.org/toolkit/docs/4.0/

[29]. http://www.cs.wisc.edu/condor/manual/v6.6.10

[30]. Anju Sharma, and Seema Bawa," An Improved Resource Discovery Approach Using P2P Model for Condor: A Grid Middleware", Proceedings Of World Academy Of Science, Engineering And Technology Volume 17, December 2006 ISSN 1307-6884

[31]. Ferstl, F. (1999) Job and resource management systems, in Buyya, R. (ed.) High Performance Cluster Computing: Architectures and Systems. Vol. 1. Upper Saddle River NJ: Prentice Hall PTR.

[32]. Rajiv Ranjan, Xingchen Chu, Carlos A. Queiroz, Aaron Harwood, Rajkumar Buyya, "A Self-Organising Federation of Alchemi Desktop Grids", Technical Report, GRIDS-TR-2007-15, Grid Computing and Distributed systems laboratory, The University of Melbourne, Australia, August 13, 2007

[33]. Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal, "Alchemi: A .NET-based Enterprise Grid Computing System", International Conference on Internet Computing 2005, pp. 269-278.

[34]. Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal, "Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework", High Performance Computing: Paradigm and Infrastructure, Laurence yang and Minyi Guo (eds), Chapter 21, pp.403-429, Wiley Press, New Jersey, USA, June 2005.

[35]. A. Iamnitchi, M. Ripeanu, and I. Foster. Locating data in (small-world?) peer-to-peer scientific collaborations. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[36]. A4 Methodology. http://www.dcs.warwick.ac.uk/research/hpsg/A4/A4.html

[37]. information about MDS available at http://www.globus.org/toolkit/about.html

[38]. The Global Grid Forum, Grid Scheduling Architecture Research Group (GSA-RG). Web site, 2005. Online: https://forge.gridforum.org/projects/gsa-rg/.

[39]. U. Schwiegelshohn, R. Yahyapour, "Resource Management for Future Generation Grids" *CoreGRID TR-0005,* May 19, 2005

[40]. Manfred Hauswirth, Roman Schmidt ,"An overlay network for Resource Discovery in Grids," http://dip.semanticweb.org/documents/An- overlay-networkfor- resource-discovery- in-Grids.pdf

[41]. Tierney, B. et al., "A Grid Monitoring Architecture," Global Grid Forum, Lemont,Illinois,U.S.A.,January2002. http://www.Gridforum.org/documents/GFD.7.pdf

[42]. Junwei Cao, Daniel P. Spooner, James D. Turner, Stephen A. Jarvis, "Agent-based Resource Management for Grid Computing", Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and Grid (CCGRID'02), 0-7695-1582-7/02, 2002.

[43]. Adriana Iamnitchi and Ian Foster, "A Peer-to-Peer Approach to Resource Location in Grid Environments," Proc. of the 11th Symposium on High Performance_Distributed Computing, Edinburgh, UK, 2002.

[44]. Iamnitchi, A., Foster, I., Nurmi, D. C., "A Peer-to-Peer Approach to Resource_Discovery in Grid Environments," Grid 2001, 2nd International workshop, Denver, CO, USA, Proc. Lecture Notes in Computer Science 2242 Springer 2001, ISBN 3-540-42949-2.

[45]. Maheswaran, M. and Krauter, K. "A Parameter-based approach to Resource Discovery in Grid computing systems", 1st IEEE/ACM

International Workshop on Grid Computing (Grid 2000), December 2000, Bangalore, India.

[46]. Huang, Y., Venkatasubramanian, N., "QoS-based Resource Discovery in Intermittently available environments," Proc. of 11th IEEE International Symposium on High Performance Distributed Computing, pp: 50 -59, HPDC-11, 2002.

[47]. I. Foster and C. Kesselman, "The Globus Project: a Status Report", In Proc. Of Seventh Heterogeneous Computing Workshop (HCW 98), IEEE Computer Society Press, March, 1998.

[48]. Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke , " The physiology of the Grid" Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[49]. Krishna Nadiminti, Akshay Luther, Rajkumar Buyya, "Alchemi: A .NETbased Enterprise Grid System and Framework" December 2005

[50]. Rajkumar Buyya and S Venugopal, "A Gentle Introduction to Grid Computing and Technologies", http://www.buyya.com/papers/GridIntroCSI2005.pdf.

[51]. S. Ludwig, P. Santen, "A Grid Service Discovery Matchmaker based on Ontology Description", Euroweb 2002 — The Web and the GRID: from e-science to ebusiness, 2002.

[52]. Information of Netbeans available at http://www.netbeans.org/

[53]. Java tutorials available at java.sun.com/docs

APPENDIX A

Installation of Alchemi

Setting up Computational Grid: Alchemi.NET

Alchemi is a .NET based middleware for establishing the computational grid environment. The steps necessary to realize a computational grid include:

The integration of individual software and hardware components into a combined networked resource.

Steps for Installing Computational Grid: Alchemi.NET

These are the steps for installing various Alchemi.NET Components. For every component we are giving its installation, configuration and operations are defined.

1. Alchemi.NET Manager Installation

The Alchemi.NET Manager can be installed in two modes

- As a normal Windows desktop application
- As a windows service. (Supported only on Windows NT/2000/XP/2003)

To install the Manager as a windows application, use the Manager Setup installer. For service mode installation use the Manager Service Setup. The configuration steps are the same for both modes. In case of the service-mode, the "Alchemi.NET Manager Service" installed and configured to run automatically on Windows start-up. After installation, the standard Windows service control Manager can be used to control the service. Alternatively the Alchemi.NET Manager Service Controller program can be used. The Manager Service controller is a graphical interface, which is exactly similar to the normal Manager application.

Install the Manager via the Manager installer. Use the sa password noted previously to install the database during the installation.

Configuration & Operation

The Manager can be run from the desktop or Start -> Programs -> Alchemi.NET -> Manager ->

Alchemi.NET Manager. The database configuration settings used during installation automatically appear when the Manager is first started.

Click the "Start" button to start the Manager.

When closed, the Manager is minimized to the system tray.

2. Cross Platform Manager Installation

Install the XPManager web service via the Cross Platform Manager installer.

Configuration

If the XPManager is installed on a different machine that the Manager, or if the default port of the Manager is changed, the web service's configuration must be modified. The XPManager is configured via he ASP.NET Web.config file located in the installation directory (wwwroot\Alchemi.NET\CrossPlatformManager by default):

```
<appSettings>
<add key="ManagerUri" value="tcp://localhost:9000/Alchemi.NET_Node" />
</appSettings>
```

Operation

The XPManager web service URL is of the format

http://[host_name]/[installation_path]

The default is therefore

http://[host_name]/Alchemi.NET/CrossPlatformManager

The web service interfaces with the Manager. The Manager must therefore be running and started for the web service to work.

3. Executor Installation

The Alchemi.NET Executor can be installed in two modes

- As a normal Windows desktop application
- As a windows service. (Supported only on Windows NT/2000/XP/2003)

To install the executor as a windows application, use the Executor Setup installer. For service mode installation uses the Executor Service Setup. The configuration steps are the same for both modes. In case of the service-mode, the “Alchemi.NET Executor Service” installed and configured to run automatically on Windows start-up. After installation, the standard Windows service control Manager can be used to control the

service. Alternatively the Alchemi.NET ExecutorServiceController program can be used. The Executor service controller is a graphical interface, which looks very similar to the normal Executor application.

Install the Executor via the Executor installer and follow the on-screen instructions.

Configuration & Operation

The Executor can be run from the desktop or Start -> Programs -> Alchemi.NET -> Executor -> Alchemi.NET Executor. The Executor is configured from the application itself. You need to configure 2 aspects of the Executor:

The host and port of the Manager connect to Dedicated / non-dedicated execution.A non-dedicated Executor executes grid threads on a voluntary basis (it requests threads to execute from the Manager), while a dedicated Executor is always executing grid threads (it is directly provided grid threads to execute by the Manager). A non-dedicated executor works behind firewalls and

Click the "Connect" button to connect the Executor to the Manager.

Appendix B

Installation of GT4

GT4 is downloadable from http://www.globus.org/toolkit/downloads/4.0.6/

In our lab we installed GT4 on Fedora Core 6 so we downloaded gt4.0.6-all-source-installer.tar.gz from the above mentioned site.

1. Before Installation

- Create a user named 'globus' in the local machine. If installing in the head node of the cluster and username is shared between nodes, create a common user to all the nodes.
- Create a directory 'globus-4.0.6' and set its access rights so that 'globus' user has the full control over it:

  [root@rohit ~] # mkdir /usr/local/globus-4.0.6

  [root@rohit ~] # chown globus:globus /usr/local/globus-4.0.6
- Set the basic environment variable GLOBUS_LOCATION to /usr/local/globus-4.0.6

2. Prerequisites

The following table provides the details of the packages that need to be installed before the installation of GT4.

Direct Downloads *	Website	Compulsory Requirement	Description
GNU tar,sed,Make,gcc	http://www.gnu.org	Yes	Basic tools
Ant 1.5.1+	http://ant.apache.org/	Yes	platform independent build tool
J2SE 1.4.2+	http://java.sun.com/j2se	Yes	java standard edition. this is not directly downloadable
zlib 1.1.4+	http://www.gzip.org	Yes	compression libs required by GPT
sudo	http://www.courtesan.com/sudo	Yes	a tool that provides normal users to execute super user level commands, used by GRAM
JDBC compliant database	http://www.postgresql.org	Yes	recommended is postgresql 7.1+ for RFT, CAS etc
IODBC	http://www.iodbc.org	-	for RLS(Replica Location Service)
Tomcat	http://jakarta.apache.org/tomcat/	-	optional for runtime needs
Junit	http://www.junit.org	-	Needed by ant for unit testing.
Mail server	-	-	Needed for sending and receiving certificates.

* **Direct Downloads:** These are downloads that can be attained directly by clicking the link.

2.1 Apache® Ant

Ant is a platform independent build tool required in both Windows and UNIX based platform before installing GT4. All the compilation of WSComponents is based on Ant. Junit package is needed along with Ant for some tests that may follow. If not installed, that particular test will fail.

1. Download the Binary Distribution of Ant: apache-ant-1.7.0-bin.tar.gz

2. Extract it into the folder where to install it.

3. set environment ANT_HOME to /usr/local/apache-ant-1.7.0 in .bash_profile

4. Copy only the junit.jar from Junit distribution into Ant 'lib' Folder

5. Example

root# cd /usr/local

root# tar xzf apache-ant-1.7.0-bin.tar.gz

2.2 Sun J2SE

J2SE is important for all the java based services.

1. Download J2SE Binary for your platform: j2sdk-1_4_2_16-linux-i586.rpm

2. just double click the .rpm package and it is installed in appropriate location

2. Set environment JAVA_HOME to /usr/java/j2sdk1.5.0_15 in .bash_profile

2.3 PostgreSQL

2.3.1 Installing Postgresql

PostgreSQL is an open source implementation of RDBMS system supporting TCP/IP communication.

1. Download Postgresql binaries client: postgresql-8.1.4-1.1.i386.rpm and server: postgresql-server-8.1.4-1.1.i386.rpm and install them

2. start pgsql

[root@rohit ~] # /etc/init.d/postgresql start

2.4 sudo

sudo is used to run commands that require Super User privilege by the ordinary user itself. It is the important requirement for WS-GRAM. So, configuring sudo is very important for successful execution of jobs using WS-GRAM. Sudo can be installed as follows

1. it is already present in fedora core 6 so we just need to configure it

1. Download the binary. And Extract it into appropriate folder

2. set PATH if needed

3. Type $(sudo)/sbin/visudo and add the following entry. It will open /etc/sudeors file. Don't use vi to edit this file.

```
# Globus GRAM entries
globus ALL=(username1,username2) NOPASSWD:
/opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute –g
/etc/grid-security/grid-mapfile
/opt/globus/GT4.0.0/libexec/globus-job-manager-script.pl *
globus ALL=(username1,username2) NOPASSWD:
/opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute –g
/etc/grid-security/grid-mapfile
/opt/globus/GT4.0.0/libexec/globus-gram-local-proxy-tool *
```

4. visudo itself reports if there are any error in the entries.

2.5 Perl

2.6 IODBC

2.7 xinetd

3. Building the Toolkit

1. Buid the Globus toolkit as globus user

```
[root@rohit ~] $ su globus
[globus@rohit ~] $ cd /home/globus
[globus@rohit ~] $ tar xzf gt4.0.6-all-source-installer.tar.gz
[globus@rohit ~] $ cd gt4.0.6-all-source-installer
[globus@rohit ~] $ ./configure
[globus@rohit ~] $ make
[globus@rohit ~] $ make install
```

4. Setting appropriate paths

Before getting into configuring of globus it is required that all paths are set correctly.

Sample .cshrc file:

```
set path = (/usr/local1/sudo/sbin /usr/local/mpich-
1.2.6/globus2/bin /usr/local/bin /usr/local1/sudo/bin
/usr/local/pgsql/bin /bin /usr/ccs/bin /usr/local/ssl/bin
```

```
/usr/sbin /usr/local/sbin \
. $path)
setenv JAVA_HOME /usr/local/J2SDK/j2sdk
setenv
MANPATH :/usr/local1/gt4.0.0/man:/usr/local/man:/usr/share/man:/u
sr/local1/sudo/man:/usr/local/pgsql/man
setenv JAVA_HOME /usr/local/J2SDK/j2sdk
setenv ANT_HOME /usr/local1/apache-ant
setenv CLASSPATH /usr/local1/junit:junit.jar
setenv GLOBUS_LOCATION /usr/local1/globus
setenv GLOBUS_HOSTNAME `hostname`
setenv PGDATA /usr/local1/pgsql/data
set path = ($ANT_HOME/bin $path $JAVA_HOME/bin $ANT_HOME/bin)
setenv GLOBUS_OPTIONS –Djava.security.egd=file:/dev/urandom
source $GLOBUS_LOCATION/etc/globus-user-env.csh
```

The GLOBUS_OPTIONS environment variable is used to set the container startup options that are used when globus-start-container command is used.
ANT_HOME environment variable is used to specify the ant installed location and JAVA_HOME is used to specify the root of the J2SDK installed location.
PGDATA must point to PostgreSQL data directory.
GLOBUS_HOSTNAME variable is optional and it is useful only when you have some hostname problems.

4. Setting up Security on first machine

1. Create the SimpleCA

   [globus@rohit ~] $ $GLOBUS_LOCATION/setup/globus/setup-simple-ca

   globus_simple_ca_xxxxxxxx_setup-0.18.tar.gz is created in the directoy /home/globus/.globus/

2. Make my machine trust that new CA by running the following commands as root

[globus@rohit ~] $ su root

[root@rohit~] $
$GLOBUS_LOCATION/setup/globus_simple_ca_xxxxxxxx_setup/setup-gsi –default

Notice that the hash value xxxxxxxx matches the hash value of my SimpleCA.

3. Get a hostcert for the machine

[root@rohit ~] $ source $GLOBUS_LOCATION/etc/globus-user-env.sh

[root@rohit ~] $ grid-cert-request –host 'hostname'

Note: we can contact the Simple CA at sharma@rohit

4. Sign the certificate using SimpleCA as globus

[root@rohit ~] $ su globus

[root@rohit ~] $ grid-ca-sign –in /etc/grid-security/hostcert_request.pem –out /home/globus/hostsigned.pem

we get the new signed certificate 01.pem at /home/globus/.globus/simpleCA/newcerts

5. copy signed certificate into /etc

[root@rohit ~] $ cp ~globus/hostsigned.pem /etc/grid-security/hostcert.pem

6. Create host certificate/key owned bt the globus

[root@rohit ~] $ /etc/grid-security

[root@rohit ~] $ cp hostcert.pem containercert.pem

[root@rohit ~] $ cp hostkey.pem containerkey.pem

[root@rohit ~] $ chown globus : globus container*.pem

7. get a usercert for sharma

[sharma@rohit ~] $ source $GLOBUS_LOCATION/etc/globus-usr-env.sh

[sharma@rohit ~] $ grid-cert-request

Note: contact globus simpleCA at sharma@rohit

7.1 get the certificate request to globus user so it can be signed, then send the signed cert back to sharma

[sharma@rohit~] $ cat /home/sharma/.globus/usercert_request.pem | mail globus@goel

[sharma@rohit ~] $ su globus

[globus@rohit ~] $ grid-ca-sign –in request.pem –out signed.pem

[globus@rohit ~] $ cat signed.pem | mail sharma@rohit

7.2 sharma copies the cert to the proper location

[globus@rohit ~] $ su sharma

[sharma@rohit ~] $ cp signed.pem ~/.globus/usercert.pem

8. Create grid-mapfile as root for authorization

[root@rohit ~] $ cd /etc/grid-security

[root@rohit ~] $ vim /etc/grid-security/grid-mapfile

[root@rohit ~] $ cat /etc/grid-security/grid-mapfile

“ / O=Grid/ OU=simpleCA-rohit.thapar.edu/OU=thapar.edu/ CN=sharma” sharma

5. Set up GridFTP

Secure credentials are in place now, it’s time to start a service

1. Create gridftp service

[root@rohit ~] $ cd /etc/grid-security

[root@rohit ~] $ vim /etc/xinetd.d/gridftp

[root@rohit ~] $ cat /etc/xinetd.d/gridftp

```
service gsiftp
{
```

```
instances               = 100
socket_type             = stream
wait                    = no
user                    = root
env                     += GLOBUS_LOCATION=/usr/local/globus-4.0.6

env                     += LD_LIBRARY_PATH=/usr/local/globus-4.0.6/lib


server                  = /usr/local/globus-4.0.6/sbin/globus-gridftp-server
server_args             = -i
log_on_success          += DURATION
nice                    = 10
disable                 = no
}
```

[root@rohit ~] $ vim /etc/services

[root@rohit ~] $ tail /etc/services

```
vboxd           20012/udp
binkp           24554/tcp               # binkp fidonet protocol
asp             27374/tcp               # Address Search Protocol
asp             27374/udp
dircproxy       57000/tcp               # Detachable IRC Proxy
tfido           60177/tcp               # fidonet EMSI over telnet
fido            60179/tcp               # fidonet EMSI over TCP

# Local services
gsiftp          2811/tcp
```

[root@rohit ~] $ /etc/init.d/xinetd reload

[root@rohit ~] $ netstat –an | grep 2811

2. Gridftp server is waiting for a request, so run a client and transfer a file

[root@rohit ~] $ cd /

[root@rohit ~] $ su sharma

[sharma@rohit ~] $ grid-proxy-init –verify –debug

[sharma@rohit~] $ globus-url-copy gsiftp://sharma.thapar.edu/etc/group file:///tmp/bacon.test.copy

[sharma@rohit ~] $ diff /tmp/bacon.test.copy /etc/group

6. Starting the web service container

setup an /etc/init.d entry for the webservices container

1. Create start-stop script as globus user

[root@rohit ~] $ su globus

[globus@rohit ~] $ vim $GLOBUS_LOCATION/start-stop

[globus@rohit ~] $ vim $GLOBUS_LOCATION/start-stop

```
#! /bin/sh
set -e
export GLOBUS_LOCATION=/usr/local/globus-4.0.6
export JAVA_HOME=/usr/java/j2sdk1.5.0_15/
export ANT_HOME=/usr/local/apache-ant-1.7.0

export GLOBUS_OPTIONS="-Xms256M -Xmx512M"

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

cd $GLOBUS_LOCATION
case "$1" in
   start)
      $GLOBUS_LOCATION/sbin/globus-start-container-detached -p 8443
      ;;
   stop)
      $GLOBUS_LOCATION/sbin/globus-stop-container-detached
      ;;
   *)
      echo "Usage: globus {start|stop}" >&2
```

```
      exit 1
     ;;
esac
exit 0
```

[globus@rohit ~] $ chmod +x$GLOBUS_LOCATION/start-stop

2. Create an /etc/init.d script to call the globus user's start-stop script

[globus@rohit ~] $ su root

[root@rohit ~] $ vim /etc/init.d/globus-4.0.6

[root@rohit ~] $ cat /etc/init.d/globus-4.0.6

```
#!/bin/sh -e
case "$1" in
  start)
    su - globus /usr/local/globus-4.0.6/start-stop start
    ;;
  stop)
    su - globus /usr/local/globus-4.0.6/start-stop stop
    ;;
  restart)
    $0 stop
    sleep 1
    $0 start
    ;;
  *)
    printf "Usage: $0 {start|stop|restart}\n" >&2
    exit 1
    ;;
esac
exit 0
```

[root@rohit ~] $ chmod +x /etc/init.d/globus-4.0.6

3. start the globus container

[root@rohit ~] $ /etc/init.d/globus-4.0.6 start

172.31.5.174 is server IP Address but when we run the above command its shows 127.0.0.1, its needs to be fixed that, Edit $GLOBUS_LOCATION/etc/globus_wsrf_core/server-config.wsdd and client-server-config.wsdd, add a line reading <parameter name="logicalHost" value="140.221.8.32" /> under the <globalConfiguration> section.

4. run a sample clients/services to interact with the container

[root@rohit ~] $su sharma

[sharma@rohit~] $ counter-client –s https://rohit.thapar.edu :

8443/wsrf/services/ConnterService

6. Configuring RFT

Configuring PostgreSQL needs the following steps:

1. configure the system to allow TCP/IP connections to postgres, as well as add a trust entry for our current host. First , set "listen_addresses = '*'" in the postgres configuration file.

[root@rohit ~] # vim /var/lib/pgsql/data/postmaster.conf

[root@rohit ~] #grep POSTMASTER /var/lib/pgsql/data/postmaster.conf

POSTMASTER_OPTIONS="-i"

Note: -i option is needed to accept TCP/IP connections to database.

[root@rohit ~] # vim /var/lib/pgsql/data/pg_hba.conf

[root@rohit ~] # grep rftDatabase /etc/pgsql/data/pg_hba.conf

host rftDatabase "globus" "140.221.8.31" 255.255.255.255 md5

[root@rohit ~] # /etc/init.d/postgresql restart

[root@rohit ~] # su postgres –c “createuser –P globus”

2. globus user create the rftDatabase

[globus@rohit ~] $ createdb rftDatabase

3. load the rft schema

[globus@rohit~] $ `psql -d rftDatabase -f $GLOBUS_LOCATION/share/ globus_wsrf_rft/rft_schema.sql`

4. change the password to globus user's password in jindi-config.xml

[globus@rohit~] $ `vim $GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi- config.xml`

[globus@rohit~] $ `grep -C 3 password $GLOBUS_LOCATION/etc/globus_wsrf_rft/ jndi-config.xml`

```
</parameter>
        <parameter>
          <name>
          password
          </name>
          <value>
          *****
```

5. restart the container to load the new RFT configuration

[root@rohit ~] # /etc/init.d/globus-4.0.6 restart

[root@rohit ~] # head /usr/local/globus-4.0.6/var/container.log

6. Try an RFT transfer to make sure the service is really working

[root@rohit ~] $ su sharma

[sharma@rohit ~] $ rft -h rohit.thapar.edu -f /tmp/rft.xfr

7. Setting up WS GRAM

[sharma@rohit ~] $ visudo

[root@rohit ~] $ cat /etc/sudoers

[sharma@rohit ~] $ globusrun-ws -submit -c /bin/true

List of Papers Published/Communicated

1. Rohit Sharma, Inderveer Chana, "A Comparison of Resource Discovery mechanisms in Alchemi, Globus and Condor", ICON 2008 (IEEE International Conference on Networks) to be held on 12 to 14 December 2008, New Delhi [Communicated in Conference proceedings].