Customized Grid Resource Discovery using Globus Toolkit

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Thapar University, Patiala

By:
Lokesh Shandil
(80631009)

Under the supervision of:
Damandeep Kaur

JUNE 2008

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Abstract

Grid computing involves heterogeneous resources of an organization working in collaboration to solve the problems that cannot be addressed by a single resource. Opportunistic sharing of Internet-connected resources is a low cost method for obtaining access to unprecedented-scale collections of resources.

However, basic problem for Grid users is how to discover the best resources required for the particular type of a job. Grid Resource Discovery deals with identifying the resources, which are suitable for a particular kind of a job, by providing description of the resources that are visible to the Grid users. There are various approaches using which Grid Discovery can be performed.

Web service based resource Discovery provided by Globus Toolkit is a well adopted standard in today's Grid environment. Globus Toolkit's Monitoring and Discovery System collects the resource information using inbuilt services and displays it using WebMDS. In this thesis, we extend the existing Resource Discovery techniques provided within Globus Toolkit which will enable us to customize the resource information according to the requirements based on the jobs to be run on the Grid and present it in our own format. Here we propose building up our own service on top of Globus MDS in order to process the information provided by MDS, customize it according to the requirement and display it using HTML page.

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Customized Resource Discovery using Globus Toolkit"**, 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 *Damandeep Kaur* 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.

(***Lokesh Shandil)***

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

(**Damandeep Kaur)**
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

First and foremost, I would like to express my sincere gratitude to my guide **Mrs. Damandeep Kaur**, Lecturer, Computer Science and Engineering Department for immense help, guidance, stimulating suggestions and encouragement all the time with this thesis work. This work would have not been possible without her encouragement. She always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under her supervision. The successful completion of this thesis is a direct consequence of the moral and material support extended by **Centre of Excellence in Grid Computing**, TU throughout this thesis.

I am equally grateful to **Dr. Seema Bawa** , Professor and Head, Computer Science and Engineering Department, **Dr. Maninder Singh**, Professor, Computer Science and Engineering Department, **Ms. Inderveer Chana**, Lecture, Computer Science and Engineering Department. My greatest thanks are to my parents who bestowed ability and strength in me to complete this work. I am deeply indebted to my parents and friends for their inspiration and ever encouraging moral support, which enabled me to pursue my studies.

I would also like to thank all the staff members and PhD Scholars Ms Anju Sharma and Ms. Shashi Bhanwar who were always there at the need of the hour and provided with all the help and facilities. I would also like to thank my friends Rohit Sharma and Kunal Goel for their great support during the tool installation and Vineet Behra and Satyarth Kumar Singh for helping me out in rest of my thesis work.

I am also very thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct–indirect help, cooperation, love and affection which made my stay at Thapar University memorable.

LOKESH SHANDIL
(80631009)

Table of Contents

List of Figures

List of Tables

Organization of Thesis

The chapters in the thesis are organized as follows:

Chapter 1 describes in detail what Grid computing is, how it evolved, its characteristics, Grid architecture, various middlewares and their architecture.

Chapter 2 contains the literature survey. It provides brief introduction to Web service, Grid services and its relation to Web Services, Grid Resource Discovery and how different middlewares implements resource Discovery.

Chapter 3 discusses the enhancements that can be made to existing approach of resource discovery using Globus toolkit and the ways how those enhancements can be done.

Chapter 4 includes setting up the test bed using Globus toolkit and how resource discovery can be customized using Globus Toolkit.

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

Chapter 1
Introduction

This chapter focuses on what grid computing is, its evolution and origin, and its characteristics. It also gives an insight into the architecture of the grid, service oriented architecture on which the grid architecture is based, various Grid Middlewares and their architecture.

1.1 Background

The grand vision of Grid computing is often presented as an analogy to power grids where users get access to electricity through wall sockets with no care or consideration for where or how the electricity is actually generated [1]. Grid computing, most simply stated, is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. The key to grid computing is the ability to break up work into small, independent chunks that can be worked on by different computers. This actually is a difficult task, and most scientific or business processes cannot be broken up into independent parts but for some applications that require a huge computation power and can be broken into smaller chunks, grid computing is a boon.

In today's complex world of high speed computing, computers have become extremely powerful and even home-based workstations are powerful enough for running complex applications. Most of these machines are connected to a LAN or the Internet or even a WAN, and it has been proven that less than half of a company's resource capabilities are used. There is a need for numerous complex applications which require a huge amount of computational and storage resources. Sometimes, it is seen that no single organization alone satisfies some of these aforementioned computational requirements; here comes the solution provided by Grid Computing.

The term Grid refers to a new infrastructure that builds on today's Internet and Web to enable and exploit large-scale sharing of resources within distributed, often loosely coordinated groups what are sometimes termed virtual organizations [2]. The use of unexploited resources of distant machines considerably increases computational power, enhances data storage, data recovery and supplies with further facilities.

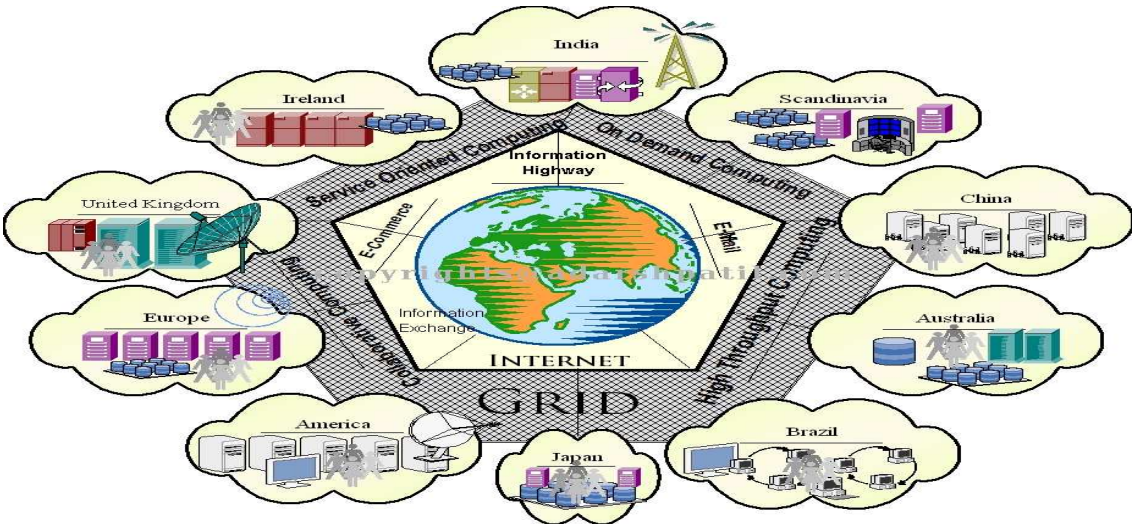

Figure 1.1: Grid Computing Overview [3]

1.2 Characteristics of Grid

The basic characteristics of Grid are as follows [4]:

- Large scale: It deals with a number of resources ranging from just a few to millions.
- Geographically distributed: Grid's resources may be located anywhere in the world.
- Heterogeneous in nature: A grid hosts both software and hardware resources that can be very varied ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, super-computers and networks.
- Resource sharing: Resources in a grid belong to many different organizations that allow several other different organizations (i.e. users) to access them.

- Different Security and Administrative policies: Each organization establishes different security and administrative policies under which their owned resources can be accessed and used.
- Resource coordination: Resources in a grid are coordinated in order to provide aggregate computing capabilities.
- Transparent access: A grid is seen as a single virtual computer so that the users don't have to bother how it works.
- Quality of Service: A grid assures the delivery of services under established Quality of Service requirements.
- Standards: A grid is built with standard services, protocols and inter-faces thus hiding the heterogeneity of the resources while allowing its scalability.
- Pervasive access**:** The grid has grant access to available resources by adapting to a dynamic environment in which resource failure is common place.

1.3 Grid Architecture

The grid architecture defines the purpose and functions of its components, while indicating how these components interact with one another [5]. The main focus of the architecture is on interoperability among resource providers and users in order to establish the sharing relationships. This interoperability, in turn, necessitates common protocols at each layer of the architectural model, which leads to the definition of a grid protocol architecture.

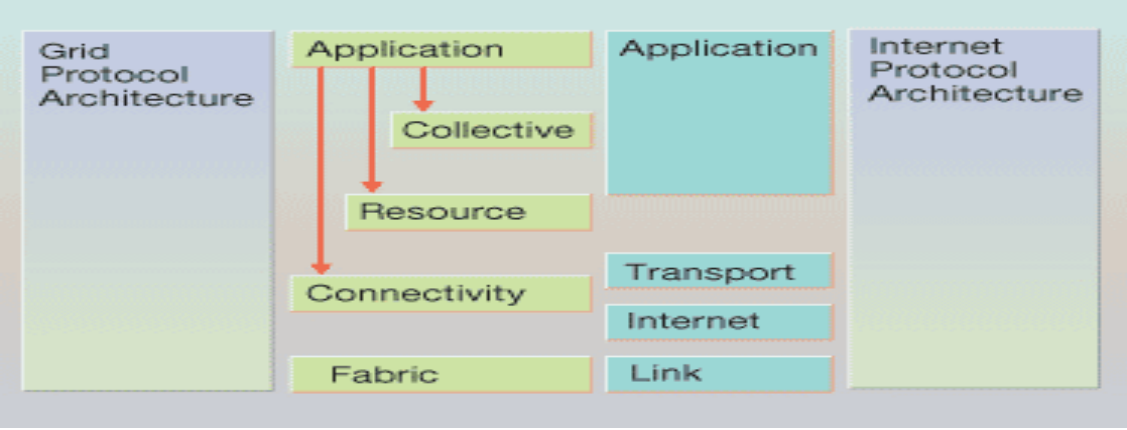

Figure 1.2: The layers of the Grid Architecture and its relationship to internet protocol architecture [6]

The protocol architecture as shown in figure 1.2 defines common mechanisms, interfaces, schema, and protocols at each layer, by which users and resources can negotiate, establish, manage, and share resources. Figure 1.2 shows the component layers of the Grid Architecture and the capabilities of each layer. The following describes [7] the core features of each of these component layers, starting from the bottom of the stack and moving upward.

- Fabric layer: the fabric layer defines the resources that can be shared. This includes computational resources, data storage, networks, catalogs, software modules, and other system resources.
- Connectivity layer: the connectivity layer defines the core communication and authentication protocols required for grid-specific networking-service transactions.
- Resource layer: this layer utilizes the communication and security protocols, defined by the network communication layer, to control secure negotiation, initiation, monitoring, accounting, and payment for the sharing of functions of individual resources.
- Collective layer: 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: the application layer enables the use of resources in a grid environment through various collaboration and resource access protocols.

1.4 Service Oriented Grid Architecture

A Service-Oriented Architecture [8] is an information technology approach or strategy in which applications make use of services available in a network. Implementing a service-oriented architecture can involve developing applications that use services, making applications available as services so that other applications can use those services, or both. A service-oriented architecture is a way of sharing functions (typically business functions) in a widespread and flexible way. SOA is defined as a loosely coupled

architecture with a set of abstractions relating to components granular enough for consumption by clients and accessible over the network with well-defined policies as dictated by the components.

The Web Services Architecture (WSA) helps to enable and define SOA, where services interact by exchanging XML. WSA is based on XML technologies such as XML Information Model [9], XML Base, and XML Schema [10]. It uses XML message exchanges, while providing the extensibility model for this message exchange pattern to adapt to various message interchange requirements (e.g., security, reliability, correlation and privacy.) These interoperable messages could be created using Simple Object Access Protocol [11] (SOAP) and SOAP extensions. SOAP extension mechanisms and XML information models are used to construct Web services extensions. Service capabilities are described using description languages such as Web Services Description Language (WSDL).

The emergence of the SOA [12] concept helps grid resources to advertise their capabilities through standard interfaces defined as part of their service extensions. This enables grid users to integrate the resources through open-standards interfaces. In addition, the operational functions of each layer in the grid architecture can be abstracted to enable easier integration across all the layers. This service abstraction of each layer of the grid architecture is shown in Figure 1.3.

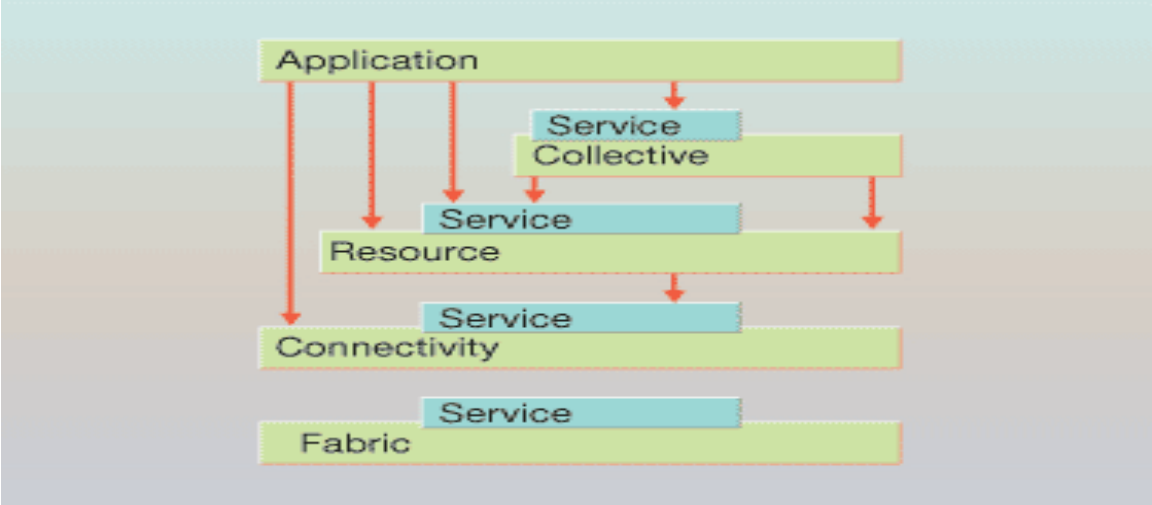

Figure 1.3: Service Oriented Grid Architecture with service interfaces

1.5 Open Grid Service Architecture

OGSA [13] is a layered architecture has a clear separation of the functionalities at each layer. The core architecture layers are the Open Grid Services Infrastructure (OGSI) and OGSA platform services. The platform services establish a set of standard services including policy, logging, service level management, and other networking services. High-level applications and services use these lower-layer platform core components to become a part of a resource-sharing grid.

The OGSA Architecture [14] is described in terms of the following capabilities:

- Infrastructure services
- Execution Management services
- Data services
- Resource Management services
- Security services
- Self-management services
- Information services

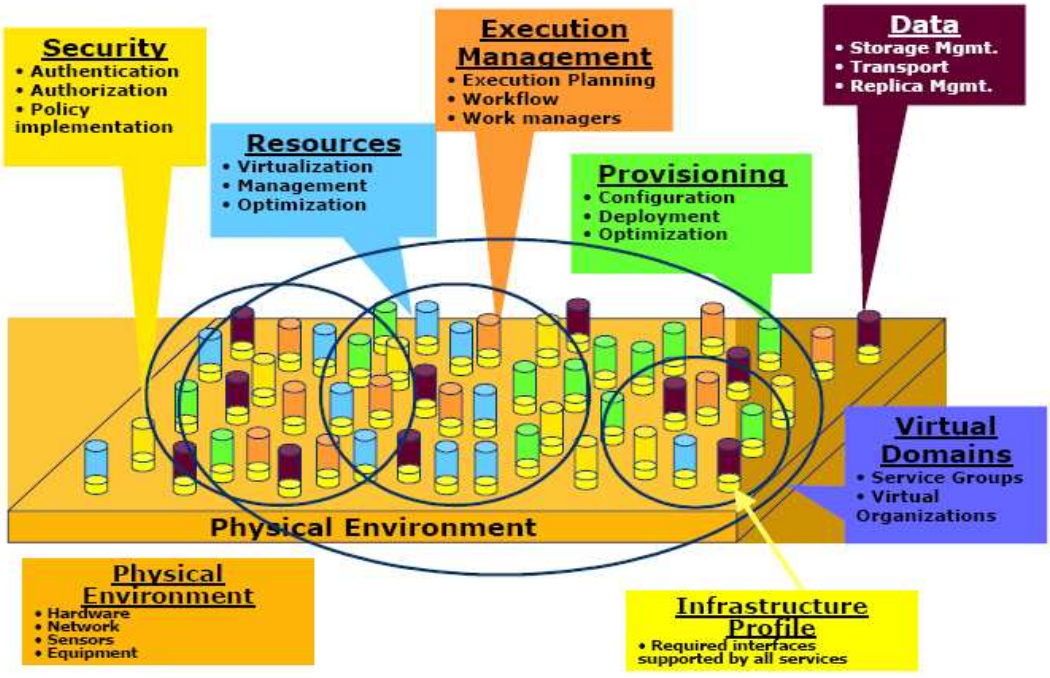

Figure 1.4: OGSA Framework [14]

1.6 Grid Middlewares

Middleware [15] is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to migrating mainframe applications to client/server applications and to providing for communication across heterogeneous platforms. There are many grid middlewares present these days designed by different organizations e.g. Globus Toolkit, Alchemi, Condor etc. All the mentioned middlewares have a different way of implementing Grid services.

1.6.1 Globus Toolkit

The open source Globus® Toolkit [16] 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.

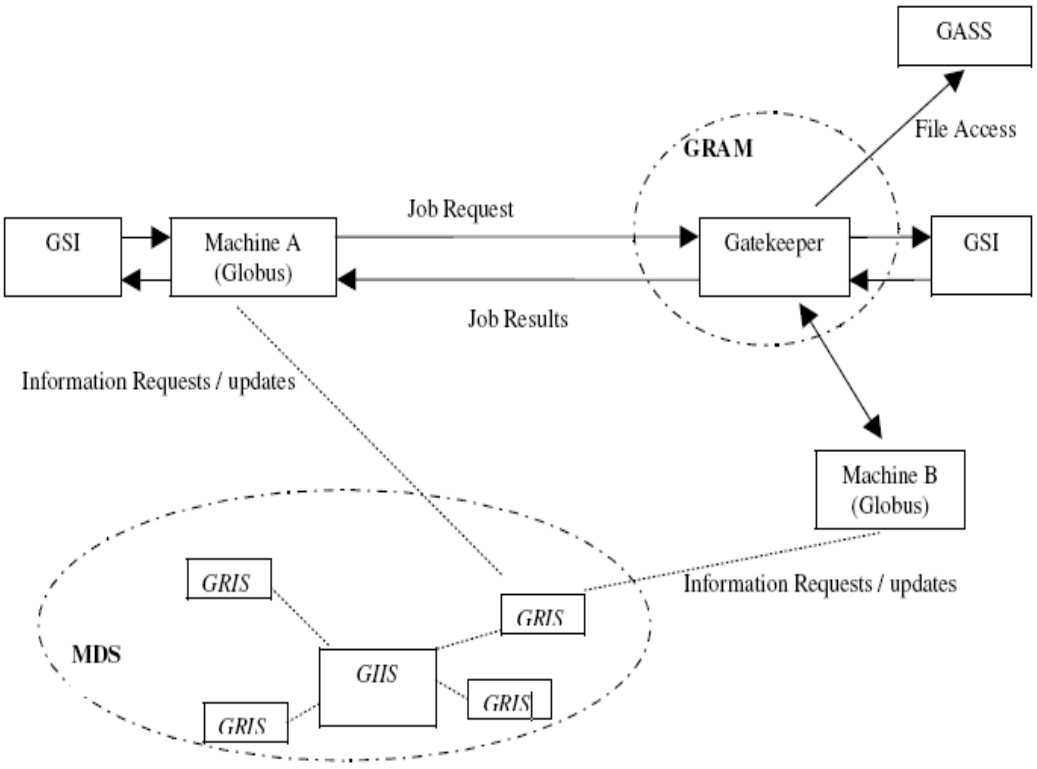

Figure 1.5: An overview of Globus Components [17]

The components of Globus are:

- GSI: The Grid Security Infrastructure controls all the security arrangements in Globus Toolkit. Security tools [18] are concerned with authentication, authorization, as well as with supporting functions such as managing user credentials and maintaining group membership information.

- MDS: The Monitoring and Discovery System is the information services component of the Globus Toolkit and provides information about the available resources on the Grid and their status. The MDS [19] is a suite of web services to monitor and discover resources and services on Grids.

- Gatekeeper: The Gatekeeper controls interactions between machine and the jobs. The job is received by the Globus Gatekeeper. Its task is to authenticate the job and its owner and to translate the request so that the Job Manager can handle it.

- GASS Server: The Global Access to Secondary Storage server [20] can be set up to provide access to other files during jobs. GASS simplifies the porting and running of applications that use file I/O to the Globus environment.

- GRAM: The Grid Resource Allocation and Management service [21] provides a single interface for requesting and using remote system resources for the execution of "jobs". The most common use of GRAM is remote job submission and control.

- GRIS: GRIS (Globus Resource Information Service) servers [17] can be located at various points across a grid. They are designed to hold information about any machine that has been registered with them.

- GIIS: All GRIS servers are registered with a separate Grid Index Information Service (GIIS) server. All the GIIS server has to store is the location of each GRIS making the request load considerably more manageable.

1.6.2 Condor

Condor [22] is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

The ClassAd mechanism in Condor provides an extremely flexible and expressive framework for matching resource requests (jobs) with resource offers (machines). Jobs can easily state both job requirements and job preferences. Likewise, machines can specify requirements and preferences about the jobs they are willing to run. These requirements and preferences can be described in powerful expressions, resulting in Condor's adaptation to nearly any desired policy. Condor's key activities [23], job-resource allocation, job startup and execution, and metadata collection and display, are kept separate, allowing compartmentalization of Condor into clearly defined components, distributed amongst submission site, central manager and execution site.

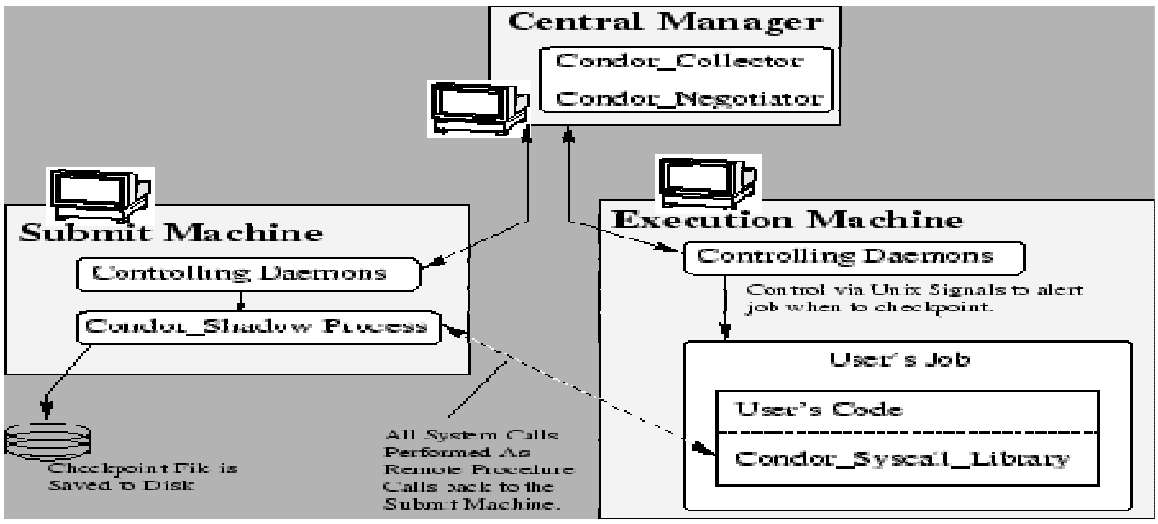

Figure 1.6: Condor Architecture Overview [23]

The components of Condor are:

- Central Manager: For every condor pool a single central manager is responsible for collecting resource characteristics and usage information from all machines in the pool and enforcing community policies.
- Submit Machine: This system client allows users to submit jobs to a local virtual 'queue'. The scheduler will request resource allocations for its jobs from the central manager during a negotiation cycle.
- Execute Machine: The execute machine, represented by the startd daemon, runs jobs on behalf of clients. It advertises its capabilities and usage information, as well as requirements and preferences upon a match, to the central manager, and manages the local execution of the job, whilst protecting resource owner policies.

1.6.3 Alchemi

Alchemi [24] is an open source software framework that allows you to painlessly aggregate the computing power of networked machines into a virtual supercomputer and to develop applications to run on the grid. It has been designed with the primary goal of being easy to use without sacrificing power and flexibility. Alchemi includes the runtime machinery to construct computational grids, a .NET API and tools to develop .NET grid applications and grid-enable legacy applications.

Alchemi Grids are constructed [25] using three types of distributed components (or nodes). These are named according to their roles with respect to a grid application. All components are installed using Windows installers. The components of Alchemi are:

- Manager: The Manager manages the execution of grid applications and provides services associated with managing thread execution. It is deployed as an executable.
- Executer: The Executor executes individual grid threads and provides services associated with executing threads. It is deployed as an executable.
- Owner: The Owner owns an application and provides services associated with the ownership of an application.

Chapter 2
Literature Review

This chapter deals with the services associated with Grid and web services concepts, Grid middlewares, Resource discovery mechanisms in different middlewares and their comparison and the motivation behind the new resource discovery technique.

2.1 Web Service Concepts

A Web service [26] is an interface that describes a collection of operations that are network-accessible through standardized XML messaging. A Web service performs a specific task or a set of tasks. A Web service is described using a standard, formal XML notation, called its service description that provides all of the details necessary to interact with the service, including message formats (that detail the operations), transport protocols, and location.

A Web service [27] is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Web Services defines techniques [28] for describing software components to be accessed, methods for accessing those components, discovery methods that enable the identification of service providers with the understanding that services are neutral entities, that is, a programming language, data, system software, a computing device, and so on.

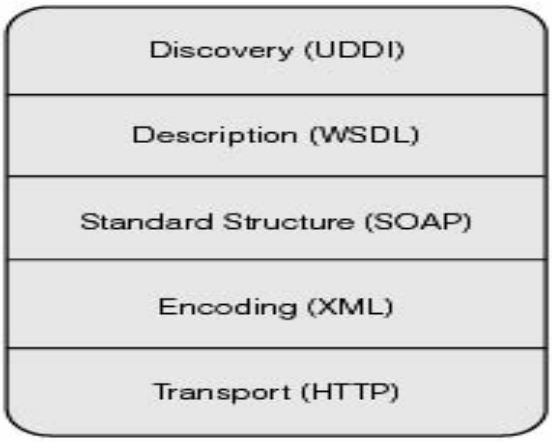

Figure 2.1: Web service Protocol Stack

The components of Web service Protocol stack are:

- Transport Protocol: responsible for transporting messages between network applications and includes protocols such as HTTP, SMTP, FTP.

- Encoding: Encoding is done in a common XML format so that the messages can be understood at either end of a network connection independent of the platform. XML [29] is a markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays. A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents. XML is a markup language for documents containing structured information.

- Standard Structure: Standard structure for message exchange is provided by SOAP. Simple Object Access Protocol is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the web services protocol stack providing a basic messaging framework upon which abstract layers can be built.

- Description: Description Protocol is used for describing the public interface to a specific web service. WSDL [30] is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

- Discovery: Discovery Protocol centralizes services into a common registry such that network web services can publish their location and description, and makes it easy to discover what services are available on the network. Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet.

2.2 Grid Services

The term "Grid service" [31] is used for a Grid resource when Grid Resource is modeled as a service as in OGSA type of architecture. In order to use web services in Grid, the resources are treated as services having the resource information as its content. A grid service is (in practice) a Web service that conforms to a particular set of coding practices, namely, a set of XML coding conventions in the form of standards. All of the Grid resources (physical or logical) in OGSA are modeled as services. These services are built on top of the SOA, and more specifically the Web Service Architecture. This enables a grid service to use the capabilities of the message model, service descriptions, and discovery. Web services standards have evolved to enable these services to extend capabilities for secure and reliable transactions. There are two core standards for utilizing grid services as web services:

2.2.1 Open Grid Service Interface

The base component of the OGSA architecture is OGSI. Based on the OGSI specification [32], a grid service instance is a Web service that conforms to a set of conventions expressed by WSDL as service interfaces, extensions, and behaviors. A grid service provides the controlled management of the distributed and often long-lived state that is

commonly required in sophisticated distributed applications. The Open Grid Services Interface defines rules about how OGSA can be implemented using Grid services that are Web services extensions. The OGSI [33] specification defines Grid services features that include statefulness, stateful interactions, the ability to create new instances, service lifetime management, notification of state changes and Grid service groups. The OGSI model requires Grid services to be specified via Grid Web Services Definition Language (GWSDL), which is an extension of WSDL.

OGSI [34] defines a component model that extends WSDL and XML Schema definition to incorporate the concepts of

- stateful Web services,
- extension of Web services interfaces,
- asynchronous notification of state change,
- references to instances of services,
- collections of service instances, and
- service state data that augments the constraint capabilities of XML Schema definition.

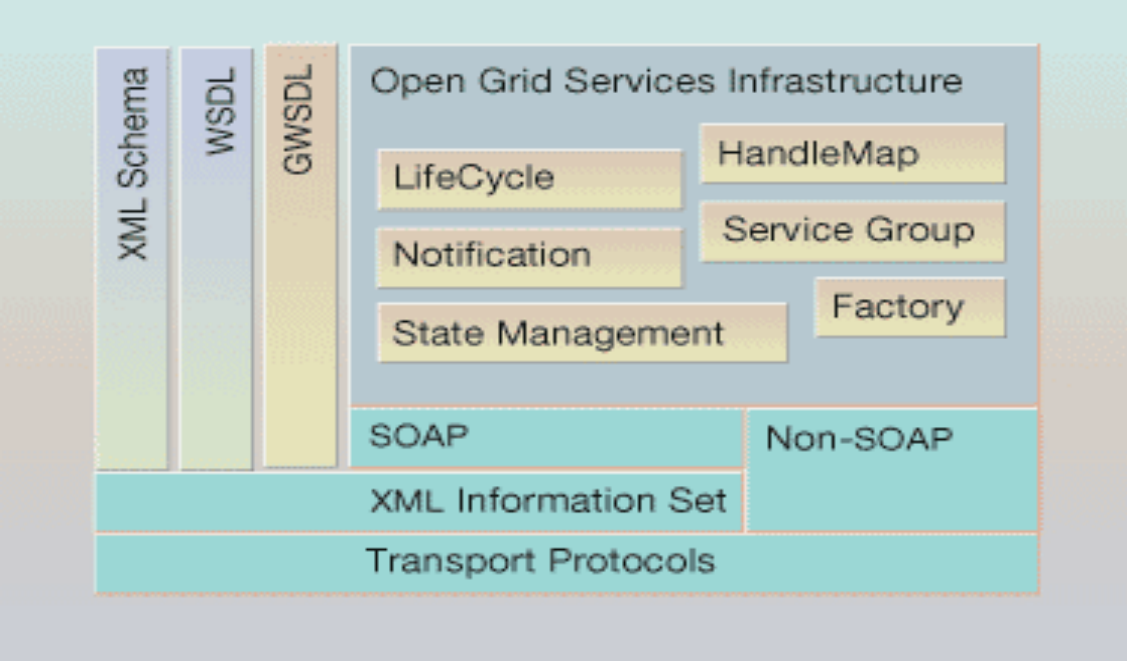

Figure 2.2: Open Grid Service Infrastructure Components [35]

Figure 2.2 shows the layering of the OGSI components in a Web service with new interfaces and behaviors. The most notable point is the extension of WSDL to provide additional state data description mechanisms. In addition to this, the specification is a set of behaviors and interfaces to support service life-cycle stages: for example, service-level management, collection management, state-change notifications, service creation mechanisms, and instance-reference management.

Message-level interoperability is a key feature of this standard, and it is achieved by using XML as the core message format and schema. One of the requirements of the services defined by OGSI is the ability to describe the concepts of state data, life-cycle properties, and instance behaviors using an OGSI description model, which is in fact a combination of WSDL and the OGSI GWSDL [36].

2.2.2 Web Service Resource Framework

Continuing evolution of Web services has made it more difficult for the Web services and Grid services to continue to merge than originally hoped. The Web Services-Resource Framework (WS-RF) provides a promising solution that can address the needs of Grid services while still holding true to the Web services foundation. WSRF is a collection of specifications to support grid services or other stateful resources and is comparable to OGSI [37].

A web service by itself is nominally stateless, i.e., it retains no data between invocations. This limits the things that can be done with web services, although workarounds exist such as having the web service read from a database, for example, or using session state by way of cookies or WS-Session.

WSRF provides a set of operations that web services may implement to become stateful. Web service clients communicate with resource services which allow data to be stored and retrieved. When clients talk to the web service they include the identifier of the specific resource that should be used inside the request, encapsulated within the WS-Addressing endpoint reference.

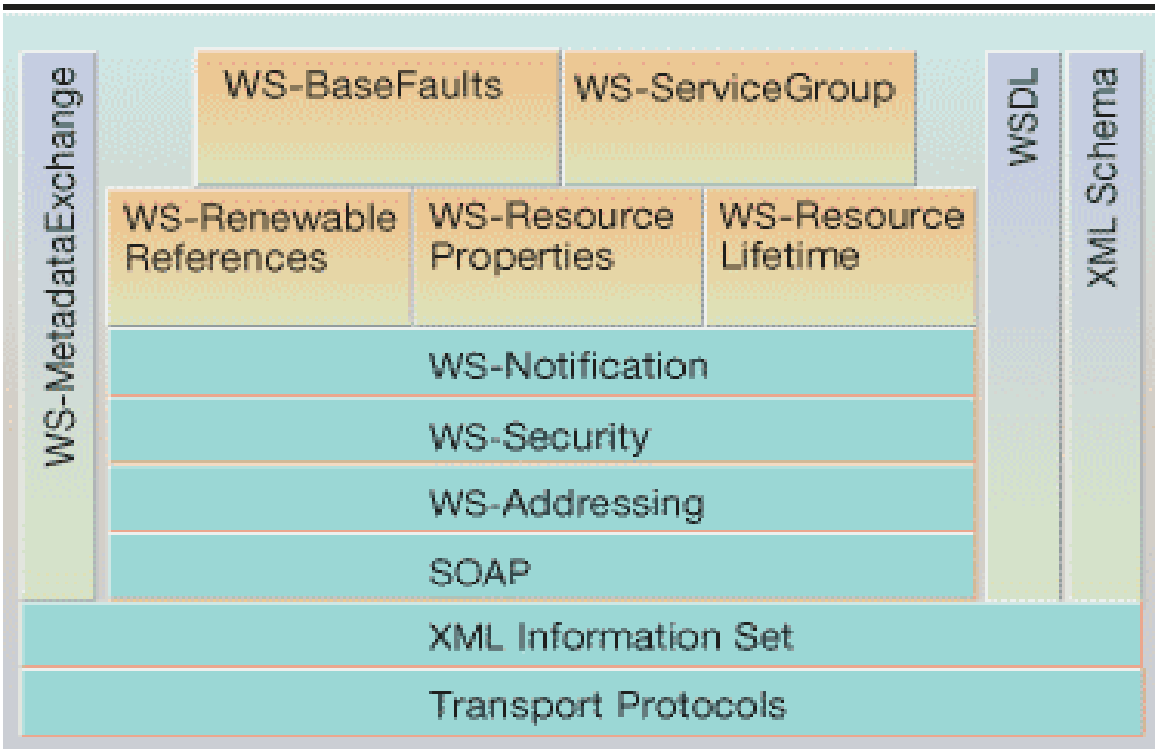

Figure 2.3: Web Service Resource Framework with WS Specifications [35]

In **Figure 2.3**, the most notable contribution is the intersection of grid computing and Web services standards and their alignment with SOA principles. This alignment will continue to help define open standards through interoperable and compatible plug-and-play service extensions to the grid architectures, thereby increasing acceptance and facilitating integration. Through this alignment with the Web services stack, grid services can use existing Web services standards, such as WS-Notification [38], WS-Addressing [39], and WS-Security [40], and build extensions for extended capabilities such as service state data, lifetime, grouping, and reference management.

The WS-Resource [1] is a construct used to model stateful resources using a Web services architecture framework. According to WSRF, a stateful resource:

- Has its state data described as an XML document
- Has a well-defined life-cycle
- Is known to and accessed by one or more Web services

The Web Services Resources Framework is a collection of four specifications [41]:

1. WS-ResourceProperties: A resource is composed of zero or more Resource Properties. WS-ResourceProperties specifies how resource properties are defined and accessed.

2. WS-ResourceLifetime: Resources have non-trivial lifecycles. In other words, they're not a static entity that is created when our server starts and destroyed when our server stops. Resources can be created and destroyed at any time. The WS-ResourceLifetime supplies some basic mechanisms to manage the lifecycle of our resources.

3. WS-ServiceGroup: The WS-ServiceGroup specifies how exactly one should go about grouping services or WS-Resources together. Although the functionality provided by this specification is very basic, it is nonetheless the base of more powerful discovery services which allow us to group different services together and access them through a single point of entry.

4. WS-BaseFaults: Finally, this specification aims to provide a standard way of reporting faults when something goes wrong during a WS-Service invocation.

2.3 Grid Resource Discovery

In the Grid, each node participating in the network makes its resources available to the other nodes. These resources are frequently changing, autonomous and heterogeneous. They can be static, like operating systems, or highly dynamic, like CPU load. In the Grid, the user gives a description of the desired services that should be satisfied by the candidate resource. The resource discovery mechanism then returns a set of the resources for the given description. Resource discovery is the process by which a node can become aware of the attributes and capabilities of other nodes in the network.

Resource Discovery deals with retrieving information about the Grid resources e.g. operating system used, total memory, processor information etc. Various grid middlewares implement resource discovery in a different way. Every resource discovery

technique has its advantages and disadvantages. Different middlewares implement Resource Discovery differently. Globus has Monitoring and Discovery System for Discovering Resources, Condor implements Matchmaking algorithm to find a best resource for the job and Resource Discovery for Alchemi can be writing a discovery service in .NET.

2.3.1 Globus MDS

The Monitoring and Discovery System [42] is a suite of web services to monitor and discover resources and services on Grids. This system allows users to discover what resources are considered part of a Virtual Organization (VO) and to monitor those resources. MDS services provide query and subscription interfaces to arbitrarily detailed resource data and a trigger interface that can be configured to take action when pre-configured trouble conditions are met.

MDS4 includes two WSRF-based services: an Index Service, which collects data from various sources and provides a query/subscription interface to that data, and a Trigger Service, which collects data from various sources and can be configured to take action based on that data.

The Index Service [43] is a registry similar to UDDI, but much more flexible. Indexes collect information and publish that information as resource properties. Clients use the standard WSRF resource property query and subscription/notification interfaces to retrieve information from an Index. Indexes can register to each other in a hierarchical fashion in order to aggregate data at several levels. Indexes are "self-cleaning"; each Index entry has a lifetime and will be removed from the Index if it is not refreshed before it expires.

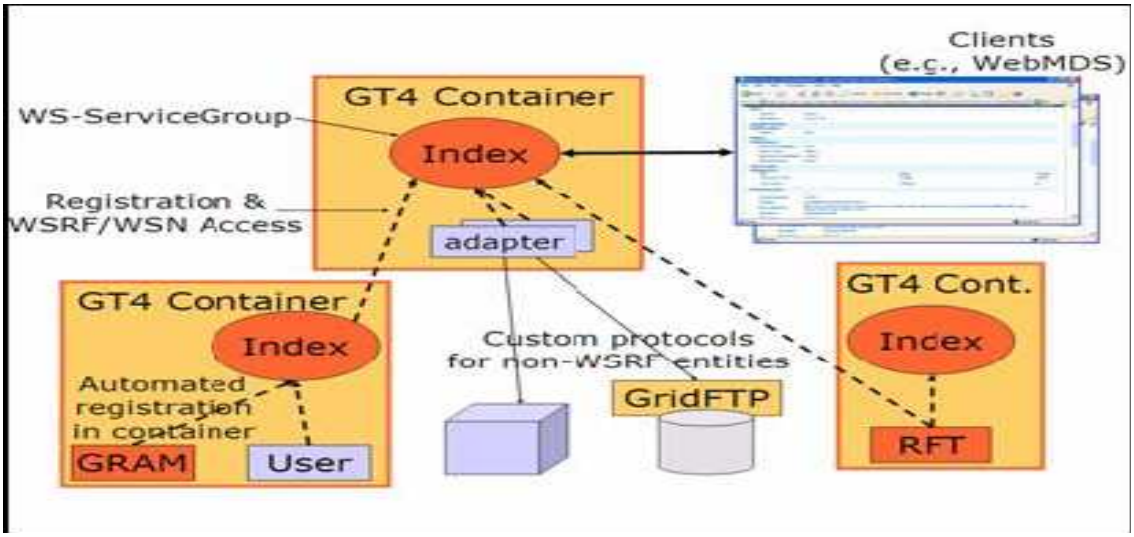

Figure 2.4: Globus MDS [44]

Each Globus container that has MDS4 installed will automatically have a default Index Service instance. The Trigger Service collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, or triggered, an action takes place, such as emailing a system administrator when the disk space on a server reaches a threshold.

2.3.2 Condor Matchmaking

Condor simplifies job submission [45] by acting as a matchmaker of ClassAds. Condor's ClassAds are analogous to the classified advertising section of the newspaper. Sellers advertise specifics about what they have to sell, hoping to attract a buyer. Buyers may advertise specifics about what they wish to purchase. Both buyers and sellers list constraints that need to be satisfied. All machines in a Condor pool advertise their attributes, such as available RAM memory, CPU type and speed, virtual memory size, current load average, along with other static and dynamic properties. This machine ClassAd also advertises under what conditions it is willing to run a Condor job and what type of job it would prefer. These policy attributes can reflect the individual terms and preferences by which all the different owners have graciously allowed their machine to be part of the Condor pool.

The grid matchmaking process involves three types of entities [46] or agents: the consumers called requesters, the producers called providers, and the matchmaking service. The matchmaking services mediate among the providers and the requesters and use a matching algorithm to evaluate a matching function that returns the matching degree. The first step in making resources available to the grid community is to advertise them. The description of a resource is known as resource advertisement, or simply resource.

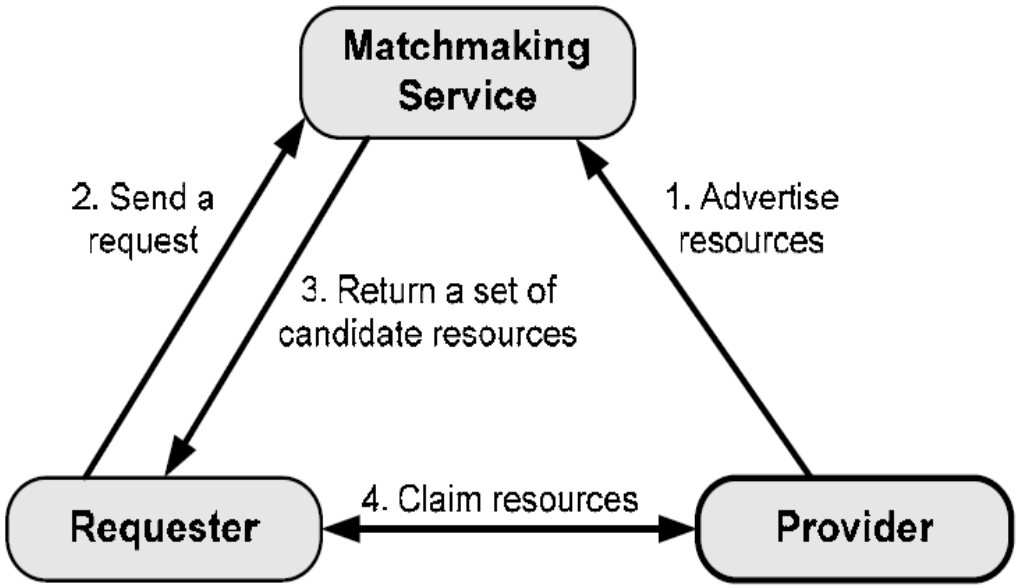

Figure 2.5: Condor Matchmaking Process

The four-step grid matchmaking process [46]:

1) Providers send resource descriptions to the matchmaking service;
2) A request is sent to the matchmaking service;
3) The matchmaking service executes a matchmaking algorithm and returns a set of ranked resources to the requester;
4) The requester chooses a resource from the set the contacts the corresponding resource provider.

2.3.3 Alchemi Resource Discovery

There is no inbuilt Resource Discovery Mechanism in Alchemi. As Alchemi is .NET based, it is dependent on Microsoft Technologies. In order to discover resources in Alchemi, .NET Web service can be written which will give the desired information about that resource. Every resource running should implement that service and send the resource information to the Alchemi Manager to be displayed or to be stored in a Database, from where it can further be used.

2.3.4 Comparison

Approach/ Middleware	**Query and Advertising Language**	**Scalability**	**Reasoning support**	**Matchmaking**	**Standard**
Globus	Resource Specification Language	High	None	No	Open, widely used
Condor	ClassAd (Classified Advertisement) Language	Low	Constraint-based reasoning	Yes	NA
Alchemi	.NET	High	None	no	Proprietary, widely used

Table 2.1: Comparison of Globus, Condor and Alchemi Middlewares

2.4 Limitations of Existing Resource Discovery Techniques

Today, there are different types of middlewares available that implement resource discovery by different means. Resource information required by the used basically depends upon the type of job to be run. For optimal utilization of resources in Grid, some unnecessary details might need to be stripped off some additional information may be added to the existing resource discovery mechanism which is inbuilt in a middleware. Stripping of the unwanted details can increase the performance of a computational resource used for resource discovery without basic loss of information. Also in other case some more additional information of resources might needed to be added for a particular type of a job e.g. Condor Matchmaking provides a way to match the jobs against the resources using a matchmaker. In this way, finding the best resource for the job is automated. But for large Grids, finding a best resource reduces performance of the Grid. Globus MDS provides the discovery information using Index Services which can be queried to get desired information about the resource. Web MDS displays the resource Information and in this way best match for a job can be found manually. This is a tedious job as the user has to search each and every resource and find the best match among them. Moreover if user is only interested in say type of Operating System then still he will see other information as well making it more tedious to find a suitable resource.

As the globus is scalable and widely used, Globus MDS can to be customized in order to incorporate the solutions to above problems. In this way, customization of an existing grid resource discovery technique helps in improving performance and provides the resource information according to the requirement

Chapter 3
Problem Formulation

This chapter emphasizes on the need of enhancements to the existing approach of Grid resource Discovery followed in Globus Toolkit and the benefits associated with customization of existing approach and how it can be done.

3.1 Web MDS and its Limitation

Monitoring and Discovering system present in Globus toolkit provides a way to discover Grid resources. MDS collects the information from all resources attached to the grid via using some inbuilt services and displays it using a user interface named WebMDS. The Service such as Default Index service contains the resource information and runs every time the Globus Container starts and gathers information from the each resource. Globus has inbuilt Information Provider that act a generator of resource information. The Information Provider can also be external e.g. Ganglia or Hawkeye etc. Globus MDS, when attached to information provider, sends a SOAP request to the information provider, which returns the discovery information as a response Information Provider sends the resource information to the default index service which in turn displays it on WebMDS. But there is a need to remove some resource information for security concerns and might need to add some additional information according to the environment and requirement. If performance is one issue, we need to gather and process only that information that is required and if usability is an issue we might need to add more resource properties. Globus MDS does not provide such customization options. Secondly we might need to store the resource information in a database to be further consumed by some application or some grid portal. Globus MDS does not have such kind of facility as well. In this thesis focus is on creating a web interface for resource discovery which will only show customized resource properties and a mechanism by which they can be stored in a database.

3.2 Proposed Approach

For customized resource discovery, we can make some enhancements to Globus MDS in order to support addition and removal of resource properties. The resource information can be extracted from Globus MDS by querying the Default Index Service present in the Globus container. The service can be used as soon as the Globus container is started. The Default Index service contains the resource information which is send by the information provider. The Index Service collects monitoring and discovery information from Grid resources, and publishes it in a single location. Information provider which generated the resource information can be internal, default with globus or external, ganglia or Hawkeye, which can be integrated with Globus Toolkit. For customizing the MDS output, a web service resource framework query is run on the Default Index Service, running on the globus Machine where discovery information of all the resources is published, and XML output is generated which contains resource information of the resources in the grid. The generated XML output is stored into some file in order to further use it. The stored xml output can be customized and displayed as HTML pages using XSLT i.e. XML Stylesheet Language Transformation and can be hosted on a web server to be visible to all the users or it can be send to a database to be used in some other application or a Grid Portal.

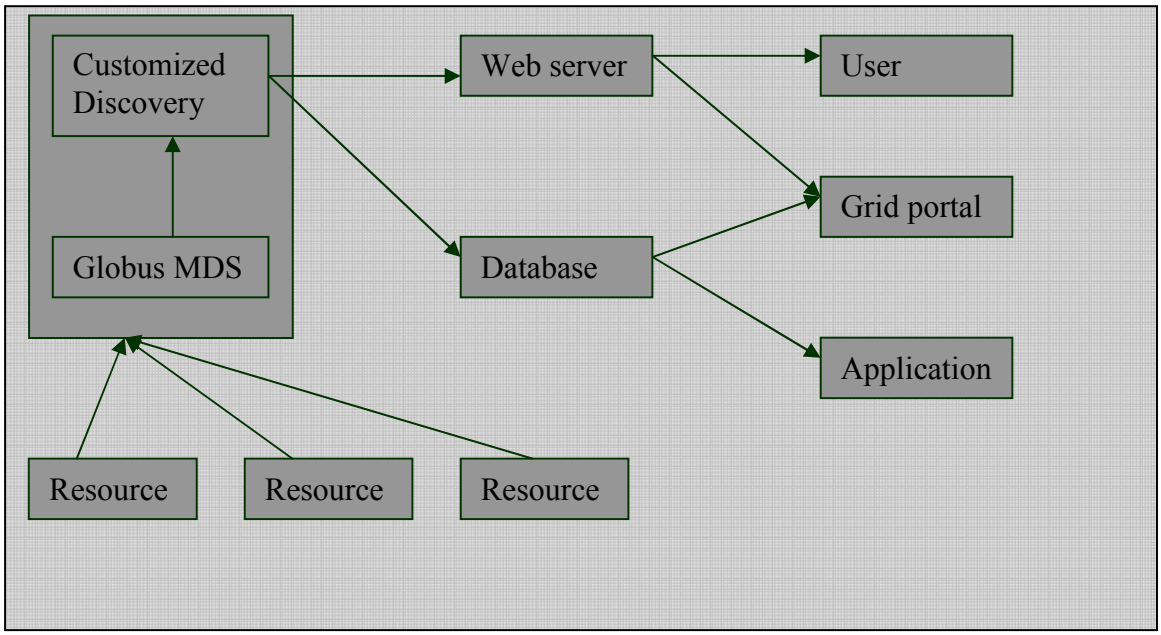

Figure 3.1: Customized Resource Discovery Overview

In order to make the resource information dynamic i.e. in order to get the updated resource information we can write a small shell script that will keep on querying the Default Index service after some specified amount of time and update the resulting XML and HTML files and display the latest resource information.

The next chapter focuses on the implementation details for setting up the test bed using Globus Toolkit, installing external Information Provider how we can customize globus MDS according to the requirement.

Chapter 4
Customized Grid Resource Discovery

This chapter contains the setting up of the test bed using Globus Toolkit 4.0, integration of Ganglia information provider with the Globus MDS, and implementation details of the Customized resource Discovery using Globus toolkit and the results involving customized Resource Discovery.

4.1 Setting up the Test Bed

Setting up the Test Bed includes installation of Globus Toolkit 4.0 on Linux systems (Fedora core 6), Apache Tomcat web server, Ganglia Information Provider and its integration with Globus Toolkit and other dependent packages which are required by above technologies.

4.1.1 Globus Toolkit Installation

A small Grid Computing Environment has been developed by installing Globus Toolkit 4.0.6 on our Linux (Fedora) machines under the Research activities carried out Centre of Excellence in Grid Computing at Thapar University, Patiala. The details of installation of Globus Toolkit are given in Appendix 1. Its components include Globus MDS, WebMDS, Grid Resource Allocation Manager, Reliable File Transfer, Grid ftp Server. Some pre-requisites like Apache Ant, Postgresql, Perl, Jakarta Tomcat, gcc C complier, and Java standard development kit. WebMDS displays the resource information accumulated by the Index service, which is present there in the Globus Container and activates as soon as Globus starts on the machine. It also involves with the setting up of Security Certificates which serves as authentication and authorization mechanism.

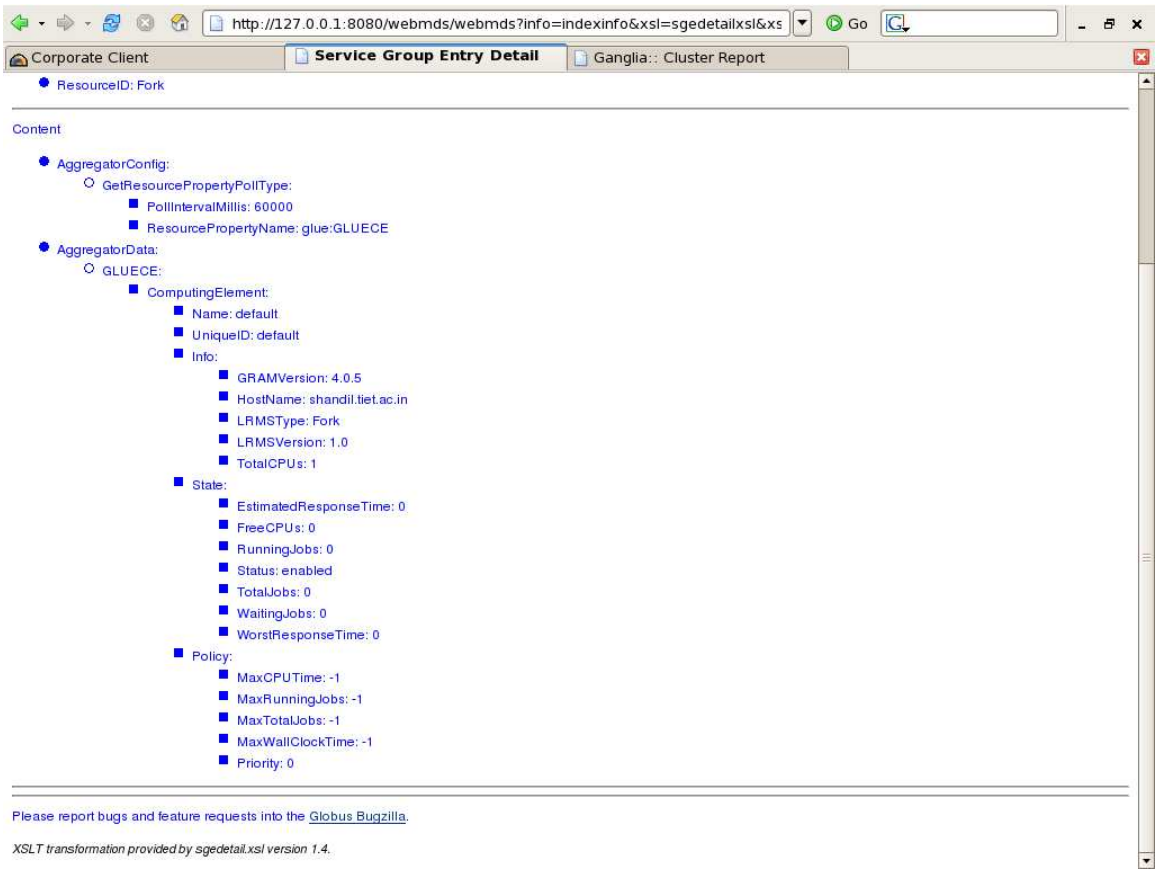

Figure 4.1: Globus WebMDS

4.1.2 Ganglia information Provider installation

Ganglia has been installed as the information provider for Globus MDS. Ganglia Monitoring Daemon (GMOND) was downloaded and installed on every machine needing to be monitored. Ganglia Meta Daemon (GMETAD) was downloaded and installed on the main machine of the cluster which is the web server or is the main machine communicating with the web server. Ganglia Web Front End has also been installed in order to check the whether Ganglia is working or not. The ganglia web front-end is the system that runs the web page for the entire monitoring system. It needs the presence of GMETAD for successful running.

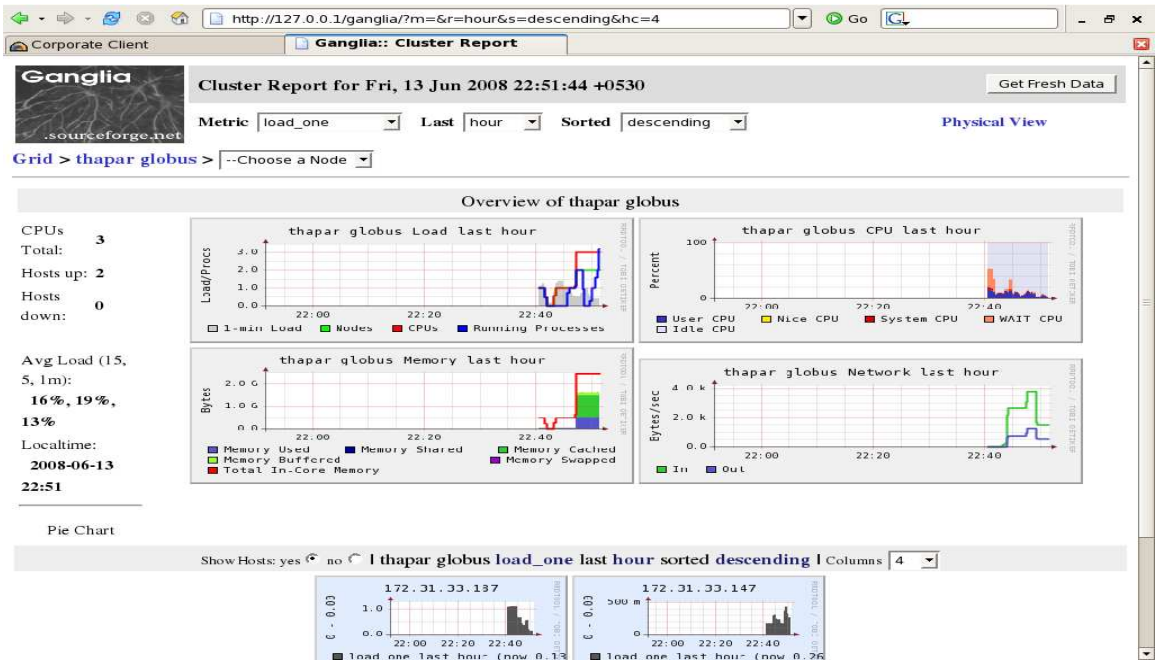

Figure 4.2: Ganglia Information provider Web Front End

4.2 Integrating Ganglia with Globus

Globus comes with default Information Provider which gives basic information about the resource which excluded information like Total memory, Processor Capability etc. So an external information Provider was required to be attached with Globus MDS in order for it give additional information.

4.2.1 Enable the RPProvider framework

The External Information Provider is part of the RPProvider Framework and is used for gathering arbitrary XML information about a resource by executing an external script. It's possible that Globus Toolkit installation already has the RPProvider framework enabled. But if it is not, a following tag is needed to be added in a server-config.wsdd file placed in $GLOBUS_LOCATION/etc/globus_wsrf_mds_index in the DefaultIndexService element, as here an external service provider is being used.

```
<parameter name="rpProviderConfigFile"
    value="/$GLOBUS_LOCATION/etc/globus_wsrf_mds_usefulrp/gluece-rpprovider-
    config.xml"/>
```

4.2.2 Changing the Default service provider

Globus used the Default Information provider which comes with the package. As Ganglia is being used as an information Provider a change is to be made to the default information Provider to Ganglia. This can be achieved by changing gluerp.xml file present at $GLOBUS_LOCATION/etc/globus_wsrf_mds_usefulrp by changing the Default service provider tag as:

```
<defaultProvider>
java org.globus.mds.usefulrp.glue.GangliaElementProducer
</defaultProvider
```

4.2.3 Creating GlueCE Configuration File

GlueCE Configuration file is to be created using mds-gluerp-configure tool. A GlueCE [47] entry represents a Computing Element which is an abstraction for an entity managing computing resources exposed to the Grid. mds-gluerp-configure [48] is a simple utility tool for generating a configuration file for the GLUE resource property provider implementation. It can create a configuration with suitable default values for both cluster and scheduler information providers.

```
mds-gluerp-configure none ganglia
/$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/ganglia-config.xml
```

“none” here represents that no scheduler name is provided, ganglia represents the Cluster name and the generated configuration file ganglia-config.xml is saved into /$GLOBUS_LOCATION/etc/globus_wsrf_mds_index location.

Next, the container can be restarted and checked whether globus WebMDS is displaying the resource information provided by Ganglia.

Figure 4.3: WebMDS output after integration with ganglia

There are two hosts attached to the grid in the above figure. It displays the information of those two resources as given by ganglia information provider. In this way, additional resource information can be added by using Information Provider such as Ganglia.

4.3 Implementation of Customized Resource Discovery

In order to customize the resource information a new Web Interface is to be written where only resource information that is required can be viewed. The DefaultIndexService contains the resource information provided by ganglia information provider. A Web Service Resource Framework query can be written to the DefaultIndexService and the resultant XML file can be stored. Wsrf-query performs a query on a resource property document. By default, a simple XPath query is assumed that returns the entire resource property document.

Wsrf-query returns an XML document that contains the resource information of all the resources attached to the Grid. In order to process further that XML document needs to be saved in order to display it as per the requirement.

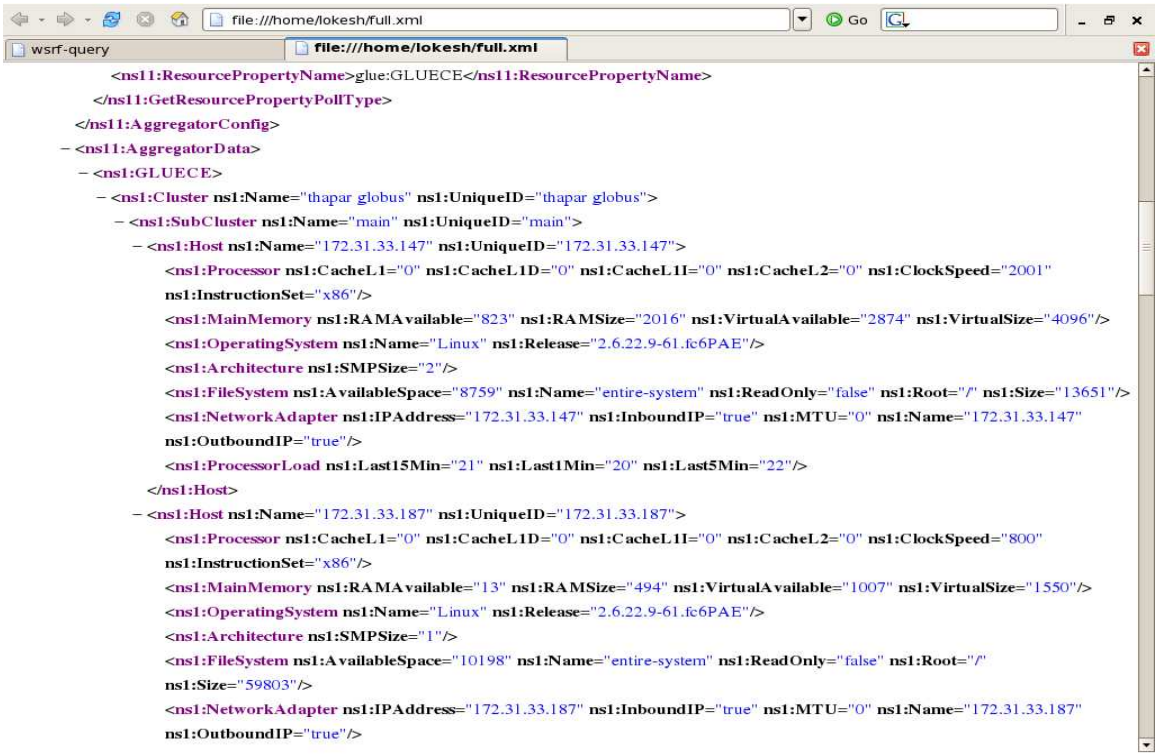

Figure 4.4: XML file generated after Querying Index service

Now data contained in an xml file can be displayed as an html page using XSLT transforms. XSL [49] stands for Extensible Stylesheet Language and it describes how the XML document should be displayed. XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X) HTML element. With XSLT, elements and attributes can be added/removed to or from the output file. One can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

To customize the resource information, XSLT is used for editing and displaying the XML file generated after querying the Index service. Only the required information is displayed and the unwanted information is cropped using XSLT. Furthermore one can design the interface according to our will. XSL documents are written according to the requirement and irrelevant information is cropped that Grid users don't require and grouped the resource information according to the type e.g. processor related information, memory details etc.

In order to convert XML document into HTML using XSL, xsltproc is used, which is a tool with LibXml for Linux which takes XML and XSLT as input and coverts it into HTML. xsltproc [50] is invoked from the command line with the name of the Stylesheet to be used followed by the name of the file or files to which the stylesheets is to be applied. The generated HTML file is hosted on the web server so that the resource information can be seen by the users in order for them to decide which resource is best for them. This can be done by saving the HTML file to ROOT folder of the Web Server.

Now in order to keep the resource information updated a check is to be made othe the resources for their state changes. Index service in Globus keeps itself updated by querying the resources after a certain amount of time. An index service can be queried over n over again in a loop for any change in the resource after a certain interval of time and update the XML and HTML files so that latest state of the resource is known to the

users. This can be done by writing a small shell script which will run itself on the background and automate the process of saving, customizing and converting the XML file into HTML and making it visible to the grid users.

In this way, the resource discovery is grouped according the type of information. The grid users can search for the type of information he is interested in and chose the best resource to submit the jobs. Thapar Customized Grid Resource Discovery provides following features.

A. Basic Thapar Grid Information

It provides basic information about the Thapar Grid and Resource Discovery and a link to basic Resource Information Page.

Figure 4.6: Main page of Customized Grid resource Discovery

B. Basic Resource Discovery information

This page provides basic information about the resource such as the names of the Cluster, sub cluster, and resources attached. It also contains the link to the particular types of information that users can be interested in like the processor related information, main memory or operating system and architecture related information, File system types and size of the resources, network adapter and recent information of the load on the processors of those resources. In this way information is grouped so at to make it easier for users to search only for the kind of information they want.

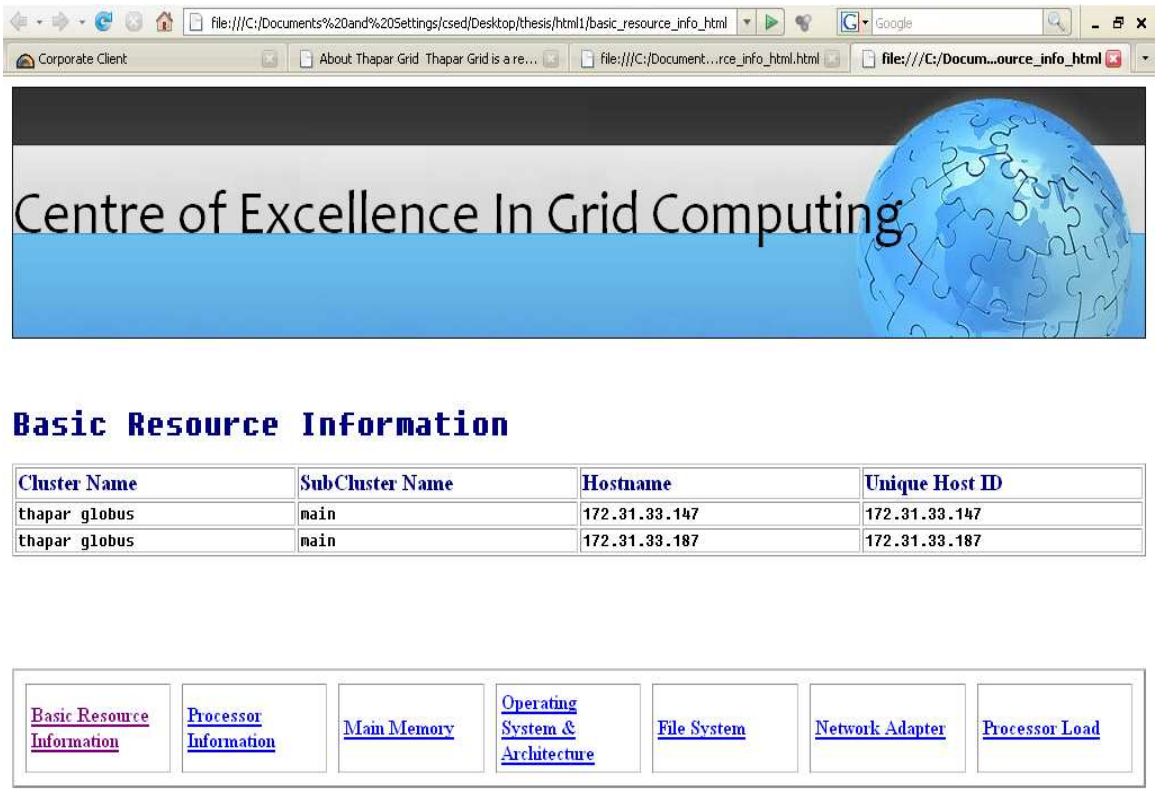

Figure 4.7: Basic Resource Information page of Customized Grid resource Discovery

C. Processor Information

This page provides the information about the processor like Cache memory, Clock Speed of the processor and the instruction set.

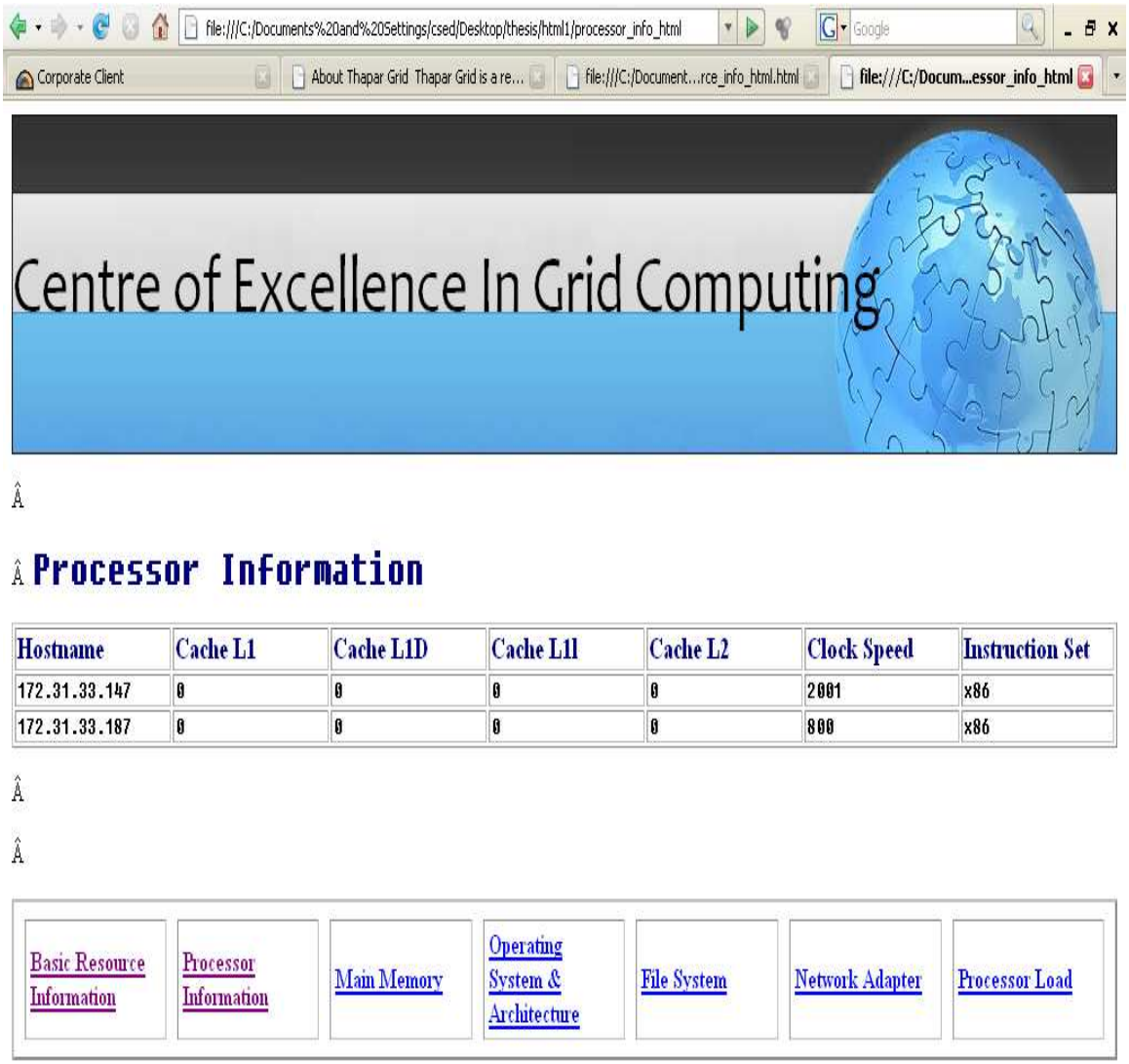

Figure 4.8: Processor Information page of Customized Grid resource Discovery

D. Memory Information

This page provides the information about the main memory like the available RAM in the system, size of the RAM, available Virtual Memory in the System and suze of that Virtual Memory.

Figure 4.9: Memory Information page of Customized Grid resource Discovery

E. Operating System and Architecture Information

This page provides the information about the Operating System of the Resource and its Architecture like the name of the Operating System, the version of the release of the Operating System and the number of Symmetric Multiprocessors that a resource is having.

Figure 4.10: Operating System and Architecture Information page of Customized Grid Resource Discovery

F. File System Information

This page provides the information about File System of the Resource like the total available space in the system, shared Read Only Memory of the resource, name of the File System of the resource and the Total Size of the File system of the resource.

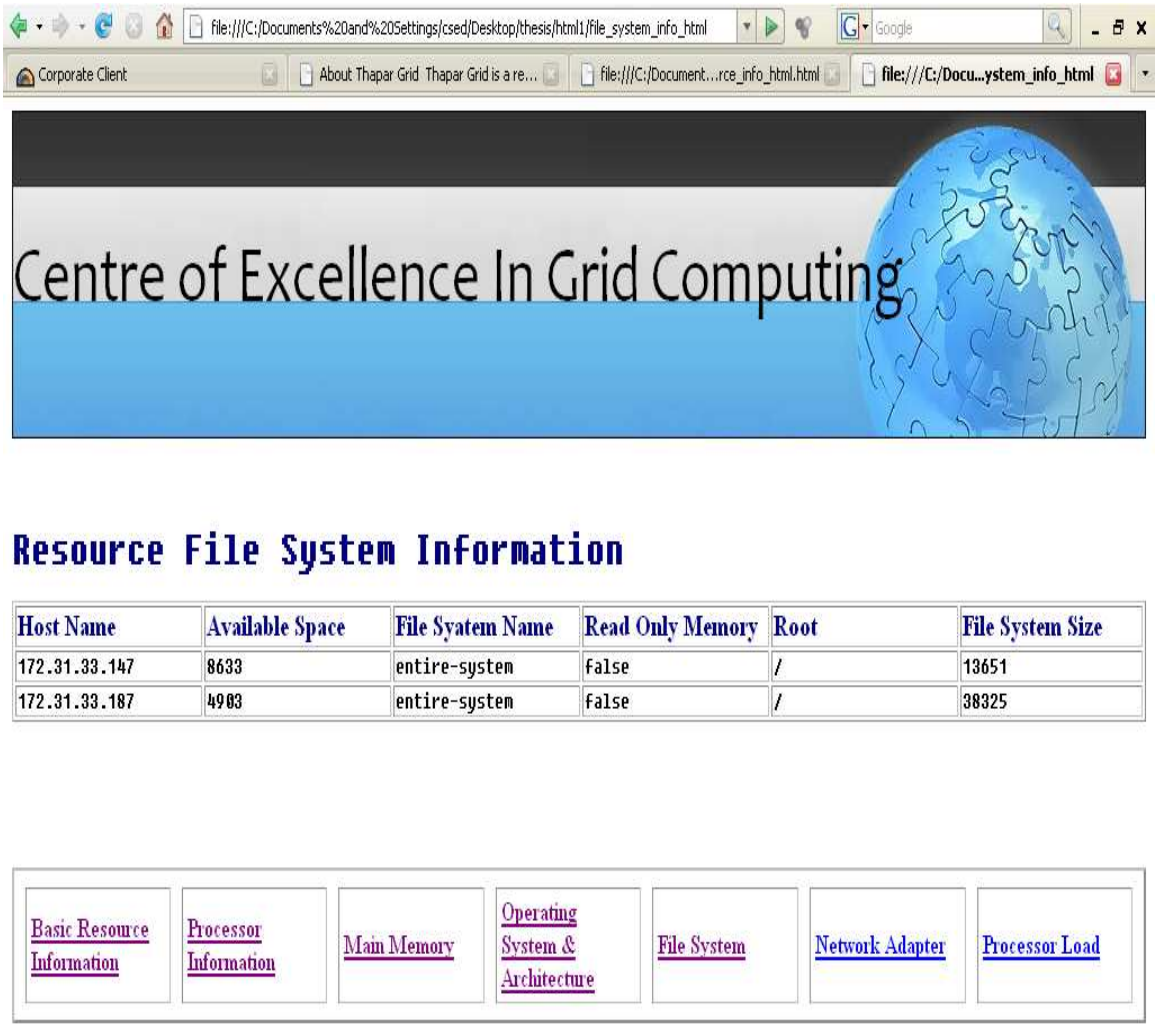

Figure 4.11: File System Information page of Customized Grid resource Discovery

G. Network Adapter Information

This page provides the information about the network adapter of the Resource like the name of the adapter, values of Inbound and Outbound IP addresses, the IP address of the resource and the value of Maximum Transmission Unit.

Figure 4.12: Network Adapter Information page of Customized Grid resource Discovery

H. Processor Load Information

This page provides the information about the load on the processor during the last 15 minutes, 5 minutes and 1 minute.

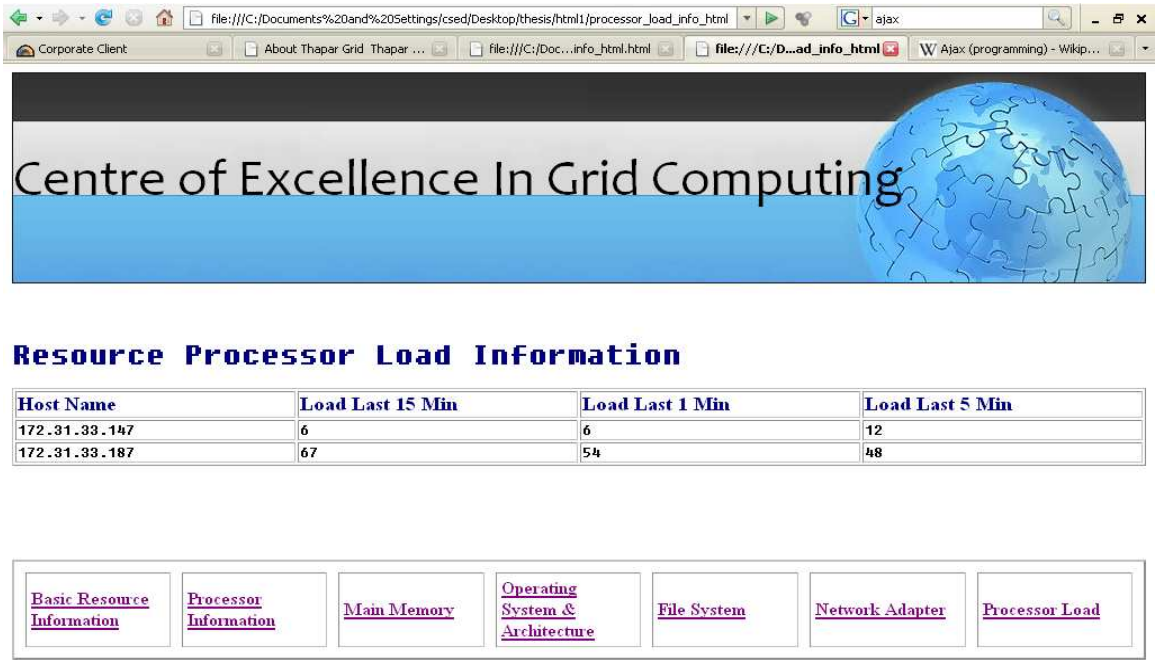

Figure 4.13: Processor Load Information page of Customized Grid resource Discovery

In this way, customized resource discovery gives the information according to the user requirements and can the information provided can be edited by changing the XSL files.

Also, a small service can then be written in order to store xml file into some database and merged with resource information provided by other Clusters so that it can be further consumed by Grid Portals and other applications.

Chapter 5
Conclusion and Future Work

This thesis work provided an insight into Grid computing, the various approaches and algorithms currently used for Resource discovery in the Grid environment, and the problems found in these approaches. Then it described design of Customized Resource Discovery Approach using Globus Toolkit as a solution to the problems in traditional approaches of Resource Discovery.

5.1 Conclusion

Discovering a Resource that satisfies a user request sufficiently is a major issue in the Grid environment due to its heterogeneous, dynamic and distributed nature. However, there is, as yet, no common standard for describing Grid resources. Since each Middleware describes a resource in its own way, information gathered from diverse sources tends to be semantically heterogeneous. Each middleware has a different way of implementing Resource Discovery and are having its advantages and disadvantages.

Globus Toolkit provides Resource Discovery solutions using Globus MDS which provides detailed information of the resources attached to it. But finding a best Resource for the job becomes difficult as one has to go through all the properties of all the resources in order to match the best resource against the job. So using Customized Grid Resource Discovery, the resource properties can be grouped according to the type, so that it is easier to look for certain type of Resource in a particular group.

Customized Grid Resource Discovery provides a simple Web Interface having details of the resources grouped according to the type e.g. Processor information, Memory information, Architecture information, File system information and Operating System information. User can select the type of information required for the job and choose the best resource. The chances of finding a best resource increases as the user is only seeing the parameters on which the job is dependent. Additional resource properties can also be added by changing the changing the associated XSL files accordingly. A different

Information Provider can be added to increase the type of information provided. In this way, using Customized Resource Discovery adds flexibility, usability to the Grid Resource Discovery process and increases the possibility of finding a best match for the job.

5.2 Summary of Contributions

1. The Customized Resource Discovery provides a user friendly Web interface which groups the information of the resources according to the type of information making it easier for users to choose the best resource for the job.

2. Only basic information which aids user to choose the best resource is provided. The irrelevant information can be stripped off.

3. The Information Providers used can also be customized to add up any additional information required by the users.

4. The resource information keeps on updating itself in a given of time. For better performance and latest results the time is calculated according to the size of the grid so that maximum number of resources updating themselves at one given time is reduced.

5.3 Future Scope

The Customized Resource Discovery presently is only applicable for Globus Toolkit only. Integrating Information Providers with other Middlewares will encourage the use the use of Customized Resource Discovery with other Middlewares as well. To gather the resource information from those middlewares a service like Default Index Service can be written. The gathered information can be displayed using Web Interface in a similar way. Also in future, the gathered resource information can also be stored into the database along with the resource information of the resources connected through other Middleware and that information in a combined form can be displayed using Web Interface.

References

[1] Bart Jacob, Michael Brown, Kentaro Fukui, Nihar Trivedi, "Introduction to Grid Computing", IBM redbooks

[2] Ian Foster, "The Grid Cluster", Cluster World Magazine, January 2004

[3] Grid Concept at "http://www.adarshpatil.com/"

[4] 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

[5] I. Foster, C. Kesselman, and S. Tuecke, The Anatomy of the Grid—Enabling Scalable Virtual Organizations, The Globus Alliance, http://www.globus.org/research/papers/anatomy.pdf.

[6] The layers of Grid Architecture at http://www.research.ibm.com/journal/sj/434/josep1.gif

[7] Jospeh, Joshy, Fellenstine Craig, "Grid Computing", Prentice Hall/IBM Press, Edition 2004 India

[8] XML Information Set, WorldWide Web Consortium (February 2004), http://www.w3.org/TR/xml-infoset/

[9] XML Information Set, WorldWide Web Consortium (February 2004), http://www.w3.org/TR/xml-infoset/

[10] XML Schema, WorldWide Web Consortium, http://www.w3.org/XML/Schema

[11] XML Protocol Working Group, WorldWide Web Consortium (2004), http://www.w3.org/2000/xp/Group

[12] Service Oriented Grid Architecture at http://www.research.ibm.com/journal/sj/434

[13] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke, "The Physiology of the Grid, An Open Grid Services Architecture for Distributed Systems Integration".

[14] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich, "The Open Grid Services Architecture, Version 1.0", GFD-I.030, 29 January 2005

[15] Middleware at http://www.sei.cmu.edu/str/descriptions /middleware.html

[16] About the Globus Toolkit at http://www.globus.org/toolkit/about.html

[17] Russell Lock, "An introduction to the Globus toolkit", 11 February 2002

[18] GT 4.0: Security at http://www.globus.org/toolkit/docs/4.0/security/

[19] Information Services (MDS): Key Concepts at http://www.globus.org/toolkit/docs/4.0/info/key-index.html

[20] Global Access to Secondary Storage at http://www.globus.org/toolkit/docs/2.4/gass

[21] WS GRAM Documentation at http://www.globus.org/toolkit/docs/3.2/gram/ws/

[22] Condor High Throughput Computing at http://www.cs.wisc.edu/condor/description.html

[23] Clovis Chapman, Paul Wilson, Todd Tannenbaum, Matthew Farrellee, Miron Livny, John Brodholt, and Wolfgang Emmerich, "Condor services for the Global Grid: Interoperability between Condor and OGSA"

[24] Alchemi - Plug & Play Desktop Grid Computing at http://www.alchemi.net/

[25] Alchemi v0.6.1 Documentation at http://www.gridbus.org/~alchemi/doc/0_6_1/index.html

[26] K. Gottschalk, S. Graham, H. Kreger, J. Snell, "Introduction to Web services architecture", IBM SYSTEMS JOURNAL, VOL 41, NO 2, 2002

[27] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard, "Web Services Architecture" , W3C Working Group Note 11 February 2004

[28] Luis Ferreira, Viktors Berstis, Jonathan Armstrong, Mike Kendzierski, Andreas Neukoetter, MasanobuTakagi, Richard Bing-Wo, Adeeb Amir, Ryo Murakawa, Olegario Hernandez, James Magowan, Norbert Bieberstein, "Introduction to Grid Computing", IBM redbooks

[29] Norman Walsh, "A Technical Introduction to XML"

[30] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note 15 March 2001

[31] Karl Czajkowski, Don Ferguson, Ian Foster, Jeff Frey, Steve Graham, Tom Maguire, David Snelling, Steve Tuecke, "From Open Grid Services Infrastructure to WSResource Framework: Refactoring & Evolution", Version 1.1, 3/05/2004

[32] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, The WS-Resource Framework, March 2004

[33] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke, "The Physiology of the Grid, An Open Grid Services Architecture for Distributed Systems Integration".

[34] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt, "Open Grid Services Infrastructure (OGSI)-Version 1.0", GWD-R (draft-ggf-ogsi-gridservice-33), June 27, 2003

[35] J. Joseph, M. Ernest, and C. Fellenstein, "Evolution of grid computing architecture and grid adoption models" IBM systems journal, Volume 43, Number 4, 2004

[36] Final OGSI Specification V1.0 (July 2003), https://forge.gridforum.org/docman2/ViewProperties.php?group_id=43&category_id=392&document_content_id=347

[37] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, The WS-Resource Framework (March 2004), http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.

[38] Publish-Subscribe Notification for Web Services (March 2004), http://www-106.ibm.com/developerworks/library/ws-pubsub

[39] A. Bosworth, D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, C. Kaler, D. Langworthy, F. Leymann, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, T. Storey, and S. Weerawarana, Web Services Addressing (WS-Addressing) (March 2003), http://msdn.microsoft.com/ws/2003/03/ws-addressing

[40] B. Atkinson, G. Della-Libera, S. Hada, M. Hondo, P. Hallam-Baker, J. Klein, B. LaMacchia, P. Leach, J. Manferdelli, H. Maruyama, A. Nadalin, N. Nagaratnam, H. Prafullchandra, J. Shewchuk, D. Simon, and C. Kaler, Web Services Security (WS-Security) (April 2002), http://www-106.ibm.com/developerworks/library/ws-secure

[41] Borja Sotomayor, "The Globus Toolkit 4 Programmer's Tutorial"

[42] Information Services (MDS) : Key Concepts at http://www.globus.org/toolkit/docs/4.0/info/key-index.html

[43] MDS4 Services at http://www.globus.org/toolkit/docs/4.0/info/key-index.html

[44] MDS4: The GT4 Monitoring and Discovery System at http://www-unix.mcs.anl.gov/~schopf/Pubs/mds4.sc.pdf

[45] Condor Matchmaking with ClassAds at http://www.cs.wisc.edu/condor/manual/v6.1/2_3Condor_Matchmaking.html

[46] Xin Bai, Han Yu, Yongchang Ji, and Dan C. Marinescu, "Resource Matching and a Matchmaking Service for an Intelligent Grid", International Journal of Computational Intelligence, Volume 1, Number 3

[47] Gluece attributes at https://twiki.grid.iu.edu/twiki/bin/view/Main/GlueCE

[48] mds-gluerp-configure at http://www.globus.org/toolkit/docs/4.0/info/usefulrp/rn01re04.html

[49] XSL Languages at http://www.w3schools.com/xsl/xsl_languages.asp

[50] xsltproc at http://gd.tuwien.ac.at/linuxcommand.org/man_pages/xsltproc1.html

List of Papers Communicated/Accepted/Published

1. Lokesh Shandil, Damandeep Kaur, "Customized Grid Resource Discovery through Stateful Web Services", GDC2008: First International Symposium on Grid and Distributed Computing, Dec 13-15 2008, Hainan Island, China. [Communicated]