Ontology based Resource Discovery Approach

in

Grid Environment

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

Master of Engineering
in
Software Engineering

By:
Santosh Kumar Jaiswal
(Roll No 8053120)

Under the supervision of

Ms. Inderveer Chana
Senior Lecturer, CSED
Thapar University, Patiala.

Dr. (Mrs.) Seema Bawa
Professor & Head, CSED
Thapar University, Patiala.

MAY 2007

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Ontology based Resource Discovery in 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 Ms. Inderveer Chana and Dr. (Mrs.) Seema Bawa.

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

(Santosh Kumar Jaiswal)

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

(Dr. Seema Bawa)
Supervisor
CSED
Thapar University
Patiala

(Ms. Inderveer Chana)
Supervisor
CSED
Thapar University
Patiala

Countersigned by

(Dr. (Mrs.) Seema Bawa)
Professor & Head
Computer Science & Engineering Department
Thapar University
Patiala

(Dr. R.K. Sharma)
Dean, Academic Affairs
Thapar University
Patiala

Acknowledgement

I wish to express my deep gratitude to Ms. Inderveer Chana, Senior Lecturer and Dr. (Mrs.) Seema Bawa Professor & Head, Computer Science & Engineering Department for providing their uncanny guidance and support throughout the preparation of the thesis report.

I am also heartily thankful to Mr. Maninder Singh, Assistant Professor, Computer Science and Engineering Department and Mrs. Shivani Goel, P.G. Coordinator, Computer Science and Engineering Department for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members, T.I.E.T. Grid Group and all my friends especially Abhishek, Anurag, Ratnesh, Puneet, Ms. Anju Sharma and Ms. Shashi who were always there at the need of the hour and provided all the help and support, which I required for the completion of the thesis.

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.

Santosh Kumar Jaiswal
(8053120)

Abstract

Grid is an emerging technology for enabling resource sharing and coordinated problem solving in dynamic multi-institutional Virtual Organizations. In the Grid environment, the resources may belong to different institutions, have different usage policies and pose different requirements on acceptable requests. One of the fundamental operations needed to support location-independent computing is Resource Discovery; it is process of locating relevant resources based on application requirements of a user.

The description of a resource is essential for automated Resource Discovery and search, selection, matching, composition and interoperation, invocation and execution monitoring; different Middleware specify different rules for describing a resource. Hence, the information gathered from these diverse sources tends to be semantically heterogeneous and needs to be correlated.

Efficient Resource Discovery needs uniform unambiguous Resource Description. To date there is no universal resource description language common to all state of the art Grid middleware. Different Grid middleware systems have different methods of resource description and it is not yet known how well these can interoperate. Hence, there is a need to utilize semantic matching of these resource descriptions.

An Ontology can provide unambiguous, asymmetric resource description that can be easily shared across various Virtual Organizations. The expressiveness of Ontology makes matchmaking process more efficient by providing list of resources based on semantics as the result of Resource Discovery process. Ontology based Resource Discovery also helps to improve Quality of Service (QoS) of Grid System and provide transparent access of Grid resources which is major goal of any Grid system

In this thesis work, a Resource Discovery Approach based on Ontology has been proposed and implemented to discover the resources similar to the ones requested by the user. Using this approach matchmaking process achieves better results than exact keyword matching prevalent in most of the Resource Discovery approaches used today.

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter focuses on a brief introduction of Grid computing. It also gives the organization of the thesis along with a brief idea about the contents of each of the chapters.

1.1 Grid Computing

Grid computing enables organizations to share computing and information resources across department and organizational boundaries in a secure, highly efficient manner. Organizations around the world are utilizing Grid computing today in such diverse areas as collaborative scientific research, drug discovery, financial risk analysis, and product design. Grid computing enables research-oriented organizations to solve problems that were infeasible to solve due to computing and data-integration constraints. Grids also reduce costs through automation and improved IT resource utilization. Finally, Grid computing can increase an organization's agility enabling more efficient business processes and greater responsiveness to change. Over time Grid computing will enable a more flexible, efficient and utility-like global computing infrastructure [1, 3]. The key to realizing the benefits of Grid computing is standardization, so that the diverse resources that make up a modern computing environment can be discovered, accessed, allocated, monitored, and in general managed as a single virtual system even when provided by different vendors and/or operated by different organizations [2].

1.2 Motivation

The first requirement for successful sharing of resources among the various organizations is an efficient discovery mechanism for locating the desired resources av1ailable in the Grid system at the time of request; however, due to the dynamic, heterogeneous and distributed nature of the Grid environment, Resource Discovery becomes challenging job.

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.

Grids are used to join various geographically distributed computational and data resources, and deliver these resources to heterogeneous user communities. These resources may belong to different institutions, have different usage policies and might be described differently depending upon Virtual Organizations [2]. In such heterogeneous, dynamic, geographical distributed environment similar type of resource can be described differently by resource consumer and resource provider and different resources can provide similar capabilities but with different quality of Resources. The resource capabilities are required to be presented in such a way that a consumer can easily discover a resource or resource ensembles with needed capabilities.

Currently in most of state of the art Grid systems, Resource discovery is done based on symmetric, attribute-based keyword matching [4]. In such systems, the values of attributes advertised by resources are compared for exact match with those required by jobs. For the comparison to be meaningful and effective, the Resource providers and consumers have to agree upon the syntax of attribute names and values beforehand so that the requestor has prior knowledge about the description of a particular Resource. However, in a Grid environment that is setup using a number of Middleware, it is difficult to enforce the syntax of Resource descriptions. Thus, there is a need to construct systems that are capable of fulfilling the requests intelligently and by “understanding” the terms involved and the relationships between them.

1.3 Organization of Thesis

The chapters in the thesis are organized as follows:

Chapter 2 describes in detail what Grid computing is, how it evolved from distributed computing, and various types of Grid systems, Grid architecture, Grid topology, and benefits of Grid.

Chapter 3 describes the Resource Discovery process and various approaches used by this process alongwih it also Resource Discovery mechanisms used in state of art Grid systems.

Chapter 4 discusses the problems found in the existing approaches to Resource Discovery and the possible solutions to these problems.

Chapter 5 includes the implementation details of the Grid setup and Ontology based Resource discovery in the Grid environment.

Chapter 6 summarizes the work presented in this thesis followed by the features that can be incorporated in future for the enhancement of the Ontology based Resource discovery in the Grid environment.

Chapter 2

Introduction to Grid Computing

This chapter describes Grid computing in detail. It explains the various related technologies like distributed computing, various kinds of Grids based on the kind of services they provide, different topologies of the Grid depending on the number of organizations that are a part of a particular Grid environment and benefits of a Grid environment.

2.1 Background

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. Furthermore, the quality and quantity requirements for some business-related advanced computing applications are also becoming more and more complex. There is a need for numerous complex scientific experiments, advanced modeling scenarios, genome matching, astronomical research, a wide variety of simulations, and complex scientific/business modeling scenarios which require a huge amount of computational and storage resources. These requirements can actually exceed the demands and availability of installed computational power within an organization. Sometimes, it is seen that no single organization alone satisfies some of these aforementioned computational requirements; here comes the solution provided by Grid Computing.

Grid computing is the process of using resources of many computers in a network to solve a single problem at a time [2]. The use of unexploited resources of distant machines considerably increases computational power, enhances data storage, data recovery and supplies with further facilities.

A well-known example of Grid computing in the public domain is the SETI [9] (Search for Extraterrestrial Intelligence) project in which thousands of people are sharing their unused processor cycles of their PCs in the vast search for signs of "rational" signals

from outer space Grid Computing uses standard protocols and open technologies that enable the correlation of servers around the planet. It can be thought of as distributed and large-scale computing cluster and as a form of network-distributed parallel processing. As long as algorithms can be divided into independent tasks, Grid Computing appears to be a promising tendency for three reasons:

- The ability to use more efficiently a given amount of computer resources
- The way to solve problems that can't be approached without an enormous amount of computing power
- The possibility for resources of many computers to be cooperatively harnessed and managed toward a common objective

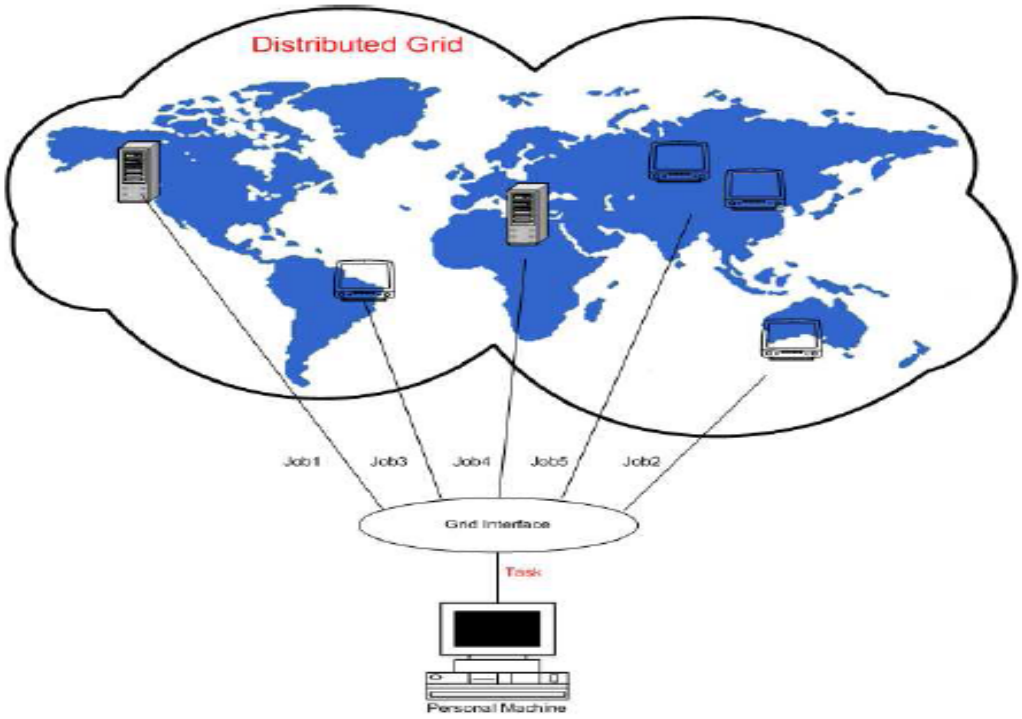


Figure 2.1 A Grid member submits a task to the Grid via the Grid Interface [10]

It undeniably creates the illusion of a simple large and powerful self-managing virtual computer, which gives the opportunity to use and reach heterogeneous systems sharing massive resources as shown in figure 2.1.

2.2 About Grid Computing

A Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities [14].

Dependable: Users need to be sure they will receive predictable and high levels of performance from components of the Grid.

Consistency : It is fundamental for the Grid to provide stable and regular services (even in presence of system heterogeneity).

Pervasive: Availability of services at any time, no matter what the environment might be.

Inexpensive: The aim of the Grid is to be broadened and thus it has to be inexpensive for members.

2.2.1 Comparison with related Technologies

There are many technologies and architectures that precede Grid Computing in terms of attempting to "merge" computing power, share storage, and facilitate program-to-program communication. Comparison of some of these architectures is presented below.

(a) Grid vs. Web

Some of the most significant differences between the Web and the Grid are the following:

- While both are operated on the Internet they are currently driven by different needs and interests: Grids by e-Science, Web by e-Commerce.
- While there is only one Web, there are many separate and distinct Grids.
- Grids have fairly limited user communities (researchers, scientists, engineers) by contrast the Web addresses and potentially serves millions of people, i.e. the general public.
- Anyone can "join" the Web; a Grid applies authentication and authorization mechanisms for accessing the Grid.
- The Grid differs from the Web in that it empowers and enables diverse resources, such as computers, data storage, instruments, application software, etc. to collaborate towards a common goal, while the Web primarily enables communication.

- Grids target processing and computations involving huge to gigantic datasets, whereas data volumes flowing across the Web are far more modest.
- There is no inherently reliable and secure data transmission on the Web reliability and security is key design issues for a Grid.

(b) Grid vs. Distributed Computing

The difference between Grid Computing and traditional distributed computing lies in computation size, scope, heterogeneity and communication. Grid Computing is distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation.

(c) Grid vs. Cluster Computing

As clusters and Grids both operate on the same underlying principle that a group of computers acts as one but there are certain differences

- In clusters, a centralized resource manager performs the resource allocation and all nodes work together cooperatively as a single unified resource. Within a Grid, each node has its own resource manager and provision of a single system view is not a goal.
- While Grids consist of heterogeneous resources, cluster computing is primarily concerned with homogeneous computational resources.
- Grid Computing integrates storage, networking, and computation resources. Clusters usually contain a single type of processor and operating system.
- Clusters typically contain a static number of processors and resources. Resources are provided and removed from the Grid on an ongoing basis.
- Clusters are homogeneous and in close physical proximity to one another, while Grids can be heterogeneous and geographically distributed.
- Clusters interconnect technology that delivers extremely low network latency, which can cause problems if clusters are not close together in proximity. Cluster and Grid Computing are complementary; many Grids incorporate clusters among the resources they manage.

2.3 Characteristics of Grids

There are many issues in computational Grid as depicted in figure 2.2 but there are three main issues that characterize computational Grids:

- **Heterogeneity**: A Grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.
  - Resources are heterogeneous
  - Resources are administratively disparate
  - Resources are geographically disparate
  - Users do not have to worry about system details (e.g., location, operating system, accounts).
  - Resources are numerous.
  - Resources have different resource management policies.
  - Resources are owned and managed by different, potentially mutually distrustful organizations and individuals that likely have different security policies and practices.
- **Scalability**: Generic definition of scalability can be given as ability of a solution to some problem (computer application or product) to continue to function well as the problem (or its context) increases in size or volume. In term of Grid it 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.
- **Dynamicity or Adaptability**: In a Grid, a resource failure is the rule, not the exception. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behavior dynamically so as to extract the maximum performance from the available resources and services.

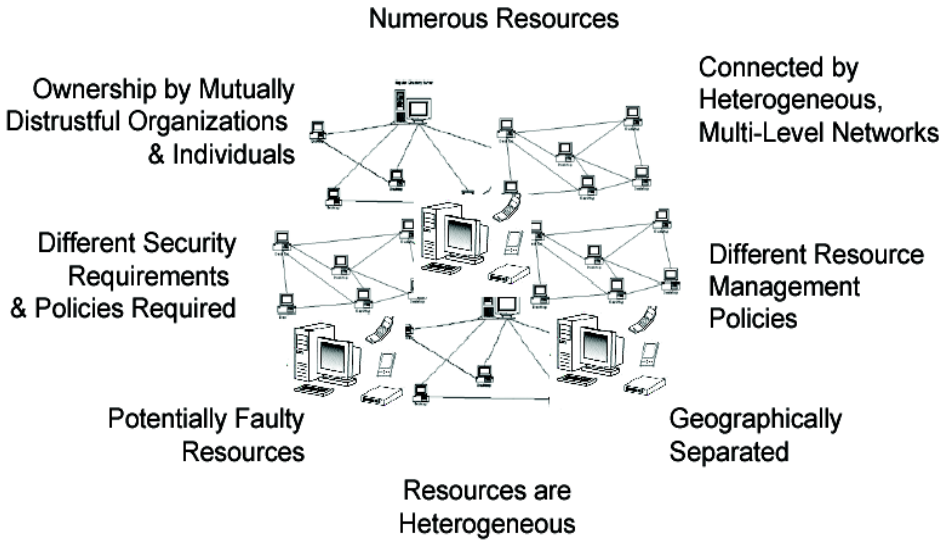


Figure 2.2 Characteristics of Grids [11]

2.4 Types of Grid

Grid research defines a number of Grid types

2.4.1 Computational Grid

Computational Grid is hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [2]. A computational Grid used to denotes hardware and software infrastructure that enables coordinated resource sharing within dynamic organizations consisting of individuals, institutions, and resources. Thus computational Grid is designed so that users won't have to worry about where scientific and engineering computations are being performed.

2.4.2 Data Grid

As Grid application areas shift from scientific computing toward industry and business applications, Data Grids, an enhancement of Computational Grids, have been designed to store, move, and manage large data sets exploited in distributed data-intensive applications [12]. The data Grid can be defined as an extension of the original Grid concept that represents a combination of large data sets, currently terabytes and will grow to petabytes, geographic distribution of users, resources and computational intensive

analysis results in complex and strict performance demands that cannot be satisfied by ordinary data infrastructure.

2.4.3 Semantic Grid

Semantic Grid is supported by two key building blocks – semantic web technologies for creating and maintaining models (ontologies), and semantic aware services and protocols for exchanging metadata. Semantic Grid is an extension of Grid and provides the infrastructure that systematically manages the complete lifecycle of metadata.

The Semantic Grid aims to provide an infrastructure within the Grid middleware – providing a core set of services and protocols to support the sharing and management of all aspects of metadata. This will enable unanticipated reuse of Grid services and resources, better support for interoperability, and flexible Grids [13, 14].

2.4.4 Knowledge Grid

The Knowledge Grid is an intelligent, sustainable Internet application environment that enables people or virtual roles (mechanisms that facilitate interoperation among users, applications, and resources) to effectively capture, publish, share, and manage explicit knowledge resources [15]. It also provides on-demand services to support innovation, cooperative teamwork, problem solving, and decision-making. It incorporates epistemology and Ontology to reflect human cognition characteristics; exploits social, ecological, and economic principles; and adopts the techniques and standards developed during work toward the next-generation Web.

2.4.5 Service Grid

Service Grids were created in order to realize the business potential of Web services. Service Grids represent distributed architectural component that realize an array of service business or utilities owning and deploying specific enabling services. Services businesses on the other hand, may either be specialized and independent businesses or revenue centers within larger enterprises offering their specialized enabling services to other enterprises [16].

2.5 Grid Architecture

Architecture principles for a Global Grid have to be established following principles summarized below:

- Employ a common network infrastructure
- Transport any traffic type
- Integration of various transport media
- Adaptation to changes
- Provide Quality of Service

The architecture of the Grid is often described in terms of “layers”, each providing a specific function [17]. In general, the higher layer is user-centric, whereas the lower layers are more hardware-centric (Figure 2.3).

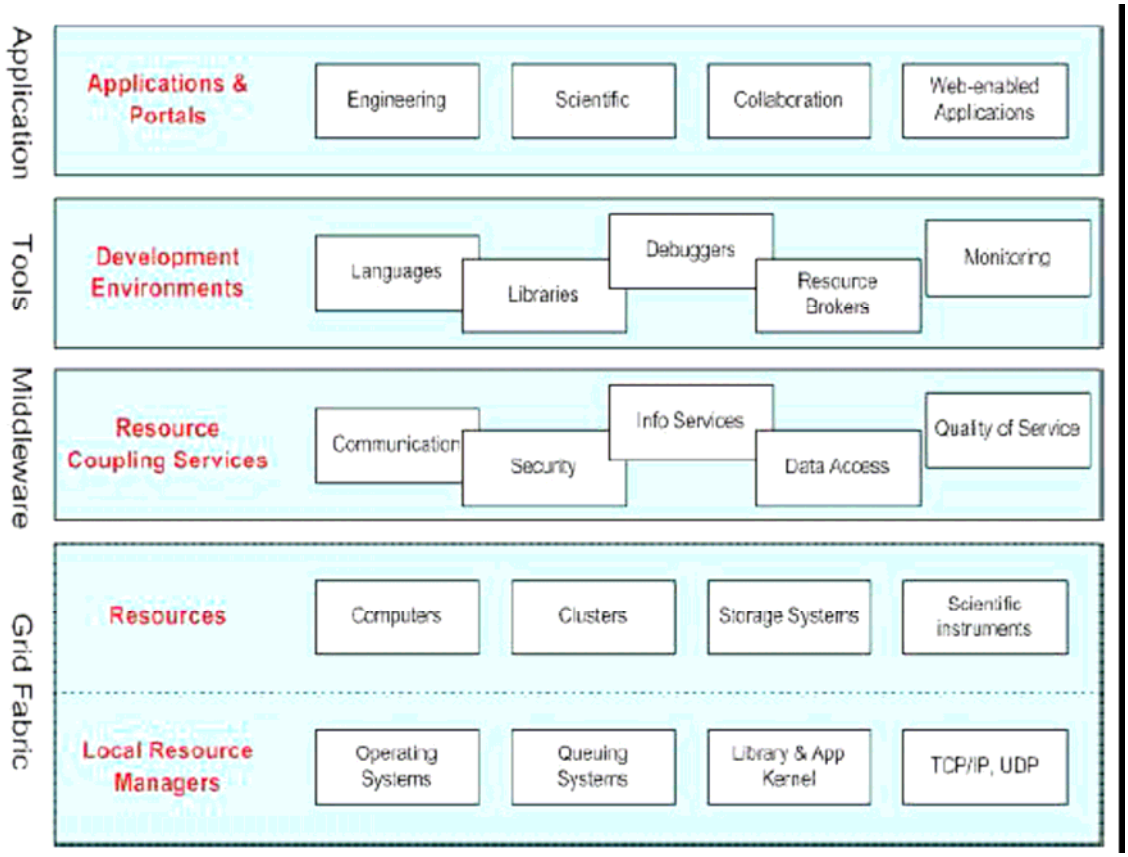


Figure 2.3 Layers of Grid Architecture [17]

2.5.1 Network Layer

At the bottom is the Network Layer, which assures the connectivity for the resources in the Grid. On top of it lies the Resource Layer, made up of the actual resources that are part of the Grid, such as computers, storage systems, electronic data catalogues, and even sensors such as telescopes or other instruments, which can be connected directly to the network.

2.5.2 Middleware Layer

The Middleware Layer provides the tools that enable the various elements (servers, storage, networks, etc.) to participate in a unified Grid environment. The Middleware layer can be thought of as the intelligence that brings the various elements together.

2.5.3 Tool Layer

The Tool Layer is made up of tools that assure connectivity between Middlewares and applications. It deals with several languages using a set of libraries to integrate the requests either from lower or higher layers.

2.5.4 Application Layer

The highest layer of the structure is the Application Layer, which includes all different user applications (science, engineering, business, financial), portals and development toolkits supporting the applications. This is the layer that users of the Grid will “see” and most of the time interacts through their browser. This layered structure can be defined in other ways. For example, the term fabric is often used for all the physical infrastructure of the Grid, including computers and the communication network. Within the Middleware layer, distinctions can be made between a layer of resource and connectivity protocols, and a higher layer of collective services. However, in all schemes, the Applications Layer remains the topmost layer.

2.6 Grid topology

Grid Architecture discussed in Figure-2.3, as software and hardware infrastructure, a network of similar machines [18]. Grid Architecture, talks about cluster, which varies in size and organization but not gives details different topologies i.e. how they are

connected. The topology of the Grid is a key factor as it determines how to manage it, to schedule jobs, to secure transactions. The simplest Grid consists of just a few machines on a local network, all of the same hardware architecture; then this model can be expanded to a local area network (between departments) with heterogeneous systems; finally, the most complicated Grid with the Internet as the communication network and heterogeneous end systems.

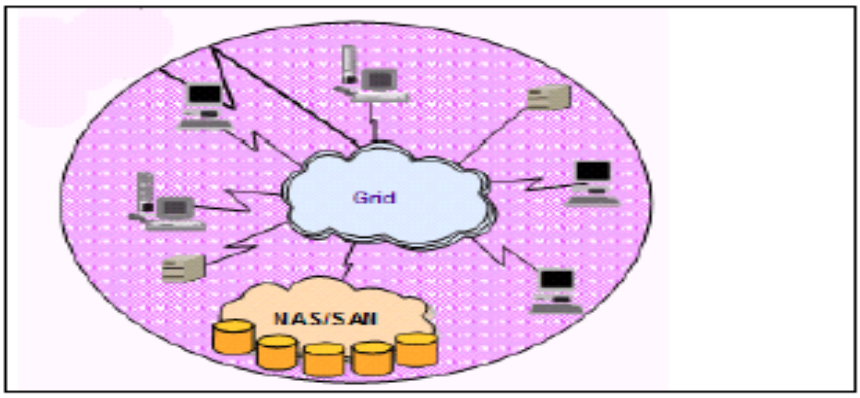


Figure 2.4. IntraGrid [18]

(a) IntraGrid (Local Area Network and homogeneous systems)

Homogeneity and proximity of nodes lessen security and policy matters. This Grid is thus a *cluster* with end systems of similar architectures and Operating Systems (OS), which ease the choice in softwares (Figure 2.4). This Grid is called as a ”software Grid” where resource sharing is fairly simple.

(b) IntraGrid (Local Area Network and heterogeneous systems)

Heterogeneity guarantees on one hand that more resources are available and on the other hand that all applications cannot be executed on all machines. Managing such Grids is of greater complexity than “software Grids” and integrity needs to be carefully looked at. The last point to mention is that security is now of interest as data from one department might need to be protected from other departments. Such a configuration is referred as an “IntraGrid” (Figure 2.5).

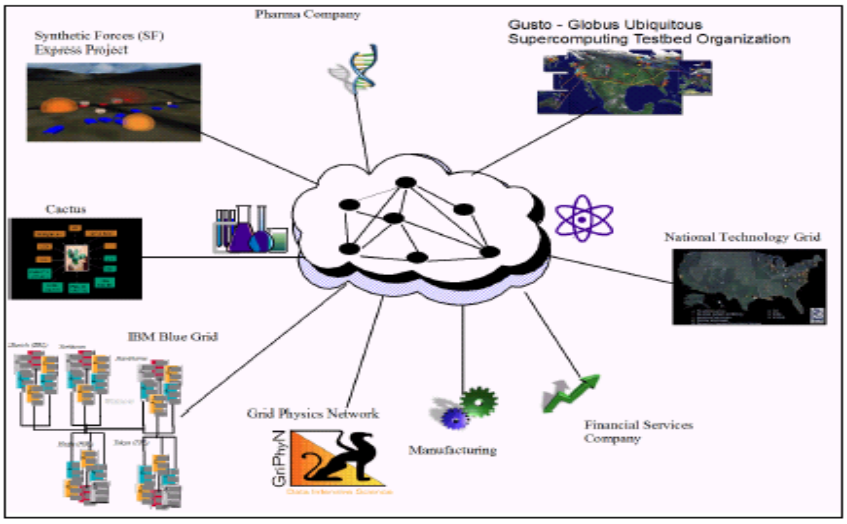


Figure 2.5. ExtraGrid [18]

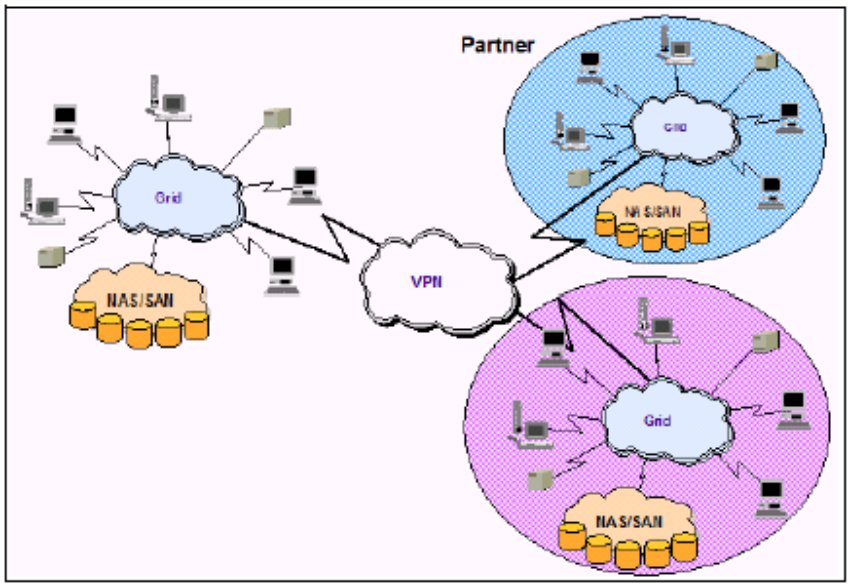


Figure 2.6. InterGrid [18]

(c) InterGrid (Internet and heterogeneous systems)

As the Grid expands to a wider area network, policies may be required, security becomes very important and the Grid needs to be hierarchically organized to reduce contention (increase scalability)(Figure 2.6). Security can be strengthened with the use of Virtual

Private Network tunneling to connect distant parts through the Internet and prevent from spying and possible attacks.

2.7 Open Standards Platform

In order to achieve true distributed resource sharing across heterogeneous and dynamic VOs [1], Grid-computing technologies require several improvements in alignment with other computing technologies. In the early days of Grid computing, a number of custom middleware solutions were created to solve the Grid problem, but this resulted in non-interoperable solutions and problematic integration among the participants.

Foster has described the Open Grid Services Architecture (OGSA) [11, 19] as a solution to the above problem. This architectural concept is a result of the alignment of existing Grid standards with emerging SOAs [20], as well as with the Web. OGSA [1, 21] provides a uniform way to describe Grid services and define a common pattern of behavior for these services.

“Open Grid Service Architecture” (OGSA) is the industry blueprint for standards-based Grid computing. “Open” refers to both the standards-development process and the standards themselves. OGSA is “service-oriented” because it delivers functionality among loosely coupled interacting services that are aligned with industry-accepted Web service standards. “Architecture” defines the components, their organizations and interactions, and the overall design philosophy.

2.7.1 OGSA Architecture

OGSA is a layered architecture, with clear separation of the functionalities at each layer. As seen in the figure, 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.

2.7.2 Semantic-OGSA (S-OGSA)

The Open Grid Service Architecture (OGSA) aims to define a core set of capabilities and behaviors for Grid systems. S-OGSA [22] is a Reference Architecture (figure 2.7) that

extends OGSA to support the explicit handling of semantics, and defines the associated knowledge services to support a spectrum of service capabilities. Guided by a set of design principles, Semantic-OGSA (S-OGSA) defines a model, the capabilities and the mechanisms for the Semantic Grid. Semantic-OGSA (S-OGSA) is guided by six general design principles these are:

(a) Parsimony of architectural elements. The architectural framework should be as lightweight as necessary and should minimize the impact on legacy Grid infrastructure and tooling.

(b) Extensibility of the framework. Rather than defining a complete and generic architecture, define an extensible and customizable one.

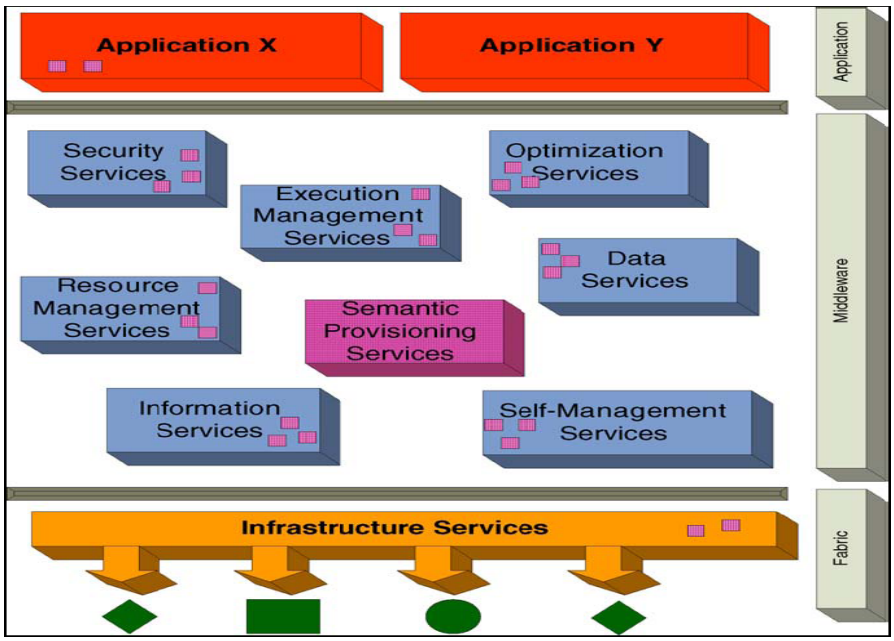


Figure 2.7 Semantic-OGSA (S-OGSA) Architecture [22]

(c) Uniformity of the mechanisms. Semantic Grids are Grids, so any S-OGSA entity included in the architecture will be OGSA-observant. OGSA observance brings about the following expectations:

(i) Similar to the Grid resources they are associated with, knowledge and metadata should exhibit manageability aspects. Semantic descriptions could have state and soft state characteristics—they have a lifetime and may change during their life.

(ii) S-OGSA must encapsulate both stateless and stateful Grid services, as OGSA does.

(iii) Knowledge services in S-OGSA are OGSA-observant Grid services.

(d) Diversity of semantic capabilities. A dynamic ecosystem of Grid services ranging over a spectrum of semantic capabilities should coexist at any one time. Grid entities do not need to be Semantic Grid entities. Semantic capability may be possible for some Grid resources all of the time, and maybe all Grid resources some of the time, not all resources all of the time.

(e) Heterogeneity of semantic representation. Any resource's property may have many different semantic descriptions, and each of them may be captured (or not) in different representational forms (text, logic, Ontology, rule).

(f) Enlightenment of services. Services should have a straightforward migration path that enables them to become knowledgeable. The cost involved in the migration to the Semantic Grid must be minimized in order to improve the impact and uptake of Semantic Grid, and to take advantage of current tooling and services

S-OGSA has three main aspects: the model i.e. the elements that it is composed of and their interrelationships, the capabilities i.e. the services needed to deal with such components and the mechanisms i.e. the elements that will enable delivery when deploying the architecture in an application.

2.8 Grid Components

Grid computing can be divided in six major components:

- User interface
- Security
- Workload management
- Scheduler
- Data management
- Resource management

2.8.1 User Interface

Accessing information on the Grid is fairly important, and the user interface component handles this task for the user. It often comes in one of two ways:

- An interface provided by an application that the user is running.

- An interface provided by the Grid administrator, much like a Web portal that provides access to the applications and resources available on the Grid in a single virtual space.

2.8.2 Security

Computers on a Grid are networked and running applications; they can also be handling sensitive or extremely valuable data, so the security component of Grid computing is of crucial concern. This component includes elements such as encryption, authentication, and authorization .The use of a user interface as a portal is also important because it can help users to learn how to query the Grid.

2.8.3 Workload management

Applications that a user wants to run on a Grid must be aware of the resources that are available; this is where a workload management service comes in handy. An application can communicate with the workload manager to discover the available resources and their status.

2.8.4 Scheduler

A scheduler is needed to locate the computers on which to run an application, and to assign the jobs required. This can be as simple as taking the next available resource, but often this task involves prioritizing job queues, managing the load, finding idle machines.

2.8.5 Data management

If an application is running on a system that doesn't hold the data the application needs, a secure, reliable data management facility takes care of moving that data to the right place across various machines, encountering various protocols.

2.8.6 Resource management

To handle such core tasks as launching jobs with specific resources, monitoring the status of those jobs, and retrieving results, a resource management facility is necessary. It's important to remember that Grid computing doesn't operate in a vacuum–just the opposite. It potentially involves every protocol and computer technology in operation today.

2.9 Benefits of Grid Computing

These are the few benefits that Grid computing provides to user community, developer community and enterprise community as well .It provides an abstraction for resource sharing and collaboration across multiple administrative domains.

Grid computing enables and manages

- Coordinated sharing and brokering of resources
- Collaborative problem-solving
- Dynamic, virtual communities.

Grid computing arose out of the need for more cost effective High Performance Computing (HPC) solutions to address critical problems in science and engineering. The initial adoption of the Grid by commercial enterprises has continued to focus on HPC because of the high return on investment and competitive advantage realized by solving compute intensive problems that were previously insolvable in a reasonable period of time or cost.

This section describes the key benefits of a Grid environment:

2.9.1 Exploit unused resources

In most organizations, computing resources are underutilized. Most desktop machines are busy less than 25% of the time (if we consider that a normal employee works 7 hours a day and that 42 hours a week and that there are 168 hours per week) and even the server machines can often be fairly idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usages. The easiest use of Grid computing would be to run an existing application on several machines. The machine on which the application is normally run might be unusually busy; the execution of the task would be delayed. Grid Computing should enable the job in question to be run on an idle machine elsewhere on the network.

2.9.2 Increase Computation

To provide users with more computational power, some crucial areas have to be considered:

(a) Hardware Improvement

Microprocessor architecture and other resource capabilities of personal computers continuously increase to provide users with additional power.

(b) Periodic Computational Needs

Some applications only need computational power once in a while. The systems are fully utilized at that times and idle the rest of the time.

(c) Capacity of Idle Machines

Machines are often idle and thus their computational power is free to use. The idea would be to use it only during that idle time and leave once the computer is in use.

(d) Sharing of Computational Results

The key to more sharing may be the development of collaboratories centers without walls, in which the nation's researchers can perform their research without regard to geographical location-interacting with colleagues, accessing instrumentation, sharing data and computational resources, and accessing information in digital libraries. Considering these areas of interest, communication networks in place could act as a medium to provide access to advanced computational capabilities, regardless of the location of resources.

This chapter described in detail what Grid computing is, evolution of Grid computing from distributed computing, various kinds of Grids based on the kind of service they provide, different topologies of the Grid depending on the number of organizations that are a part of a particular Grid environment, and benefits of Grid environment.

The next chapter discusses the role of Grid Resource management (GRM) in Grid architecture its issues, Resource Discovery process and its steps, and the various existing approaches used for Grid service discovery.

Chapter 3

Grid Resource Discovery

This chapter describes the Resource Management in Grid Environment and its issues, Resource Discovery process and steps followed in this process. This chapter also describes various approaches used for Grid Resource Discovery alongwith the comparison of Resource Discovery approaches used in existing Grid Systems.

3.1 Grid Resource Management

Grid resource management is 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 [23]. 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 users, resource owners, and administrators of the system.

While Grids have become almost commonplace, the use of good Grid resource management tools is far from ubiquitous because of the many open issues of the field [24].

The various issues to be considered while managing Grid resources include:

(a) Multiple layers of schedulers

Grid resource management involves many players and possibly several different layers of schedulers. At the highest level are Grid-level schedulers that may have a more general view of the resources but are very “far away” from the resources where the application will eventually run. At the lowest level is a local resource management system that manages a specific resource or set of resources. Other layers may be in between these, for example one to handle a set of resources specific to a project. At every level additional people and software must be considered.

(b) Lack of control over resources

Grid schedulers are not local resource management systems. Grid-level scheduler may not (usually does not) have ownership or control over the resources. Most of the time, jobs will be submitted from a higher-level Grid scheduler to a local set of resources with no more permission than the user would have. This lack of control is one of the challenges that must be addressed.

(c) Shared resources and variance

Related to the lack of control is the lack of dedicated access to the resources. Most resources in a Grid environment are shared among many users and projects. Such sharing results in a high degree of variance and unpredictability in the capacity of the resources available for use. The heterogeneous nature of the resources involved also plays a role in varied capacity.

(d) Conflicting performance goals

Grid resources are used to improve the performance of an application. Often, however, resource owners and users have different performance goals: from optimizing the performance of a single application for a specified cost goal to getting the best system throughput or minimizing response time. In addition, most resources might have local policies that must be taken into account e.g. how much of the scheduling process should be done by the system and how much by the user? What are the rules for each?

3.2 Resource Discovery

Resource Discovery is the first phase of Grid Resource Management. Resource Discovery 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 [25].

It involves the user 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.

3.2.1 Steps in Resource Discovery [21]

Most users perform Resource Discovery in three steps (Figure 3.1):

- Authorization filtering,
- Job requirement knowledge, and
- Filtering to meet the minimal job requirements.

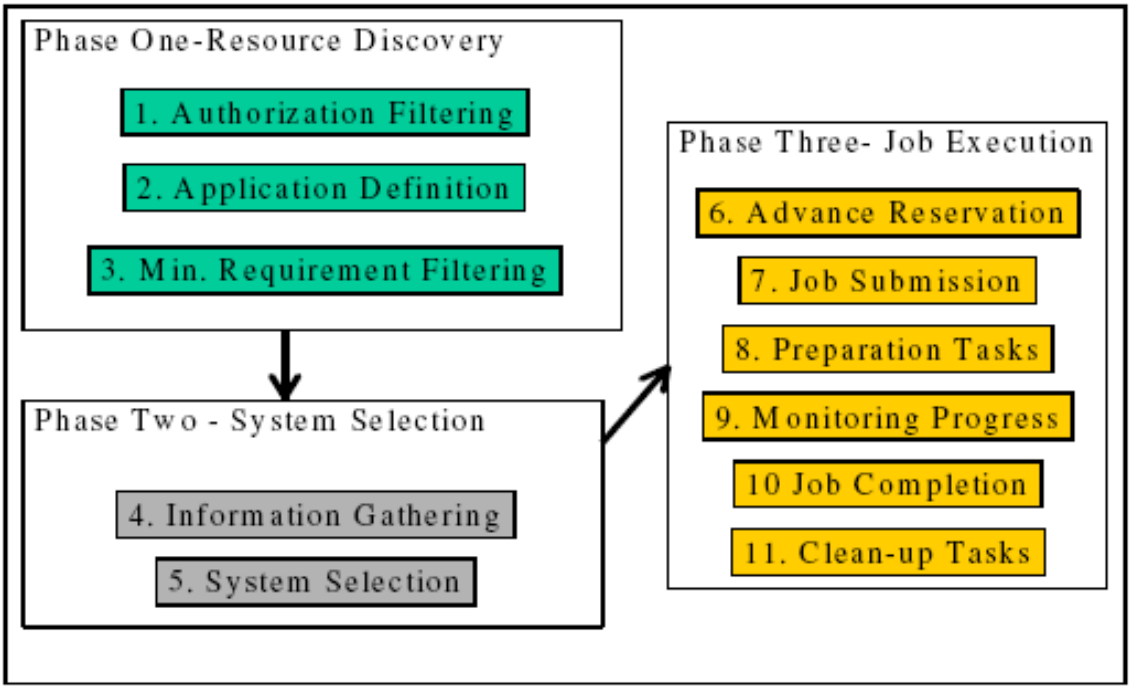


Figure 3.1 Resource Scheduling Phases [21]

Step 1: Authorization Filtering

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

Step 2: Application requirement definition

In order to proceed in Resource Discovery, the user 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 could be matched to a set of resources.

Step 3: Minimal requirement filtering

Given a set of resources to which a user 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 user 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 user can eliminate an inappropriate resource it is done at this stage to simplify the information gathering in the next phase.

3.2.2 Grid Resource Discovery problem

The Grid Resource Discovery problem can be defined as the problem of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements. The problem is complicated by the fact that some resource information (e.g., CPU load or available storage) changes dynamically [2].
The Resource Discovery problem comprises several necessary components. Iamnitchi and Foster define four axes of the solution space. In their taxonomy,

- The **membership protocol** determines how nodes are added to the Grid and start being discovered,
- **Overlay construction** determines direct collaborator pairs among members,
- **Preprocessing** describes steps that are executed prior to information requests being issued, and **request processing** maps specific requests to resource sets that can satisfy them.

3.3 Approaches to Resource Discovery

The Grid Resource Discovery process can be defined as the process of matching a query for resources described in terms of required characteristic, to a set of resources that meets

the expressed requirements. There are two approaches to Resource Discovery that can be classified as

- **Query Based**
- **Agent Based**.

In a query based discovery the resource information store is queried for resource availability. Most contemporary Grid systems follow this approach. In agent based discovery agents traverse the Grid system to gather information about resource availability [26].

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 [27,28].

3.4 Agent-Based Resource Discovery

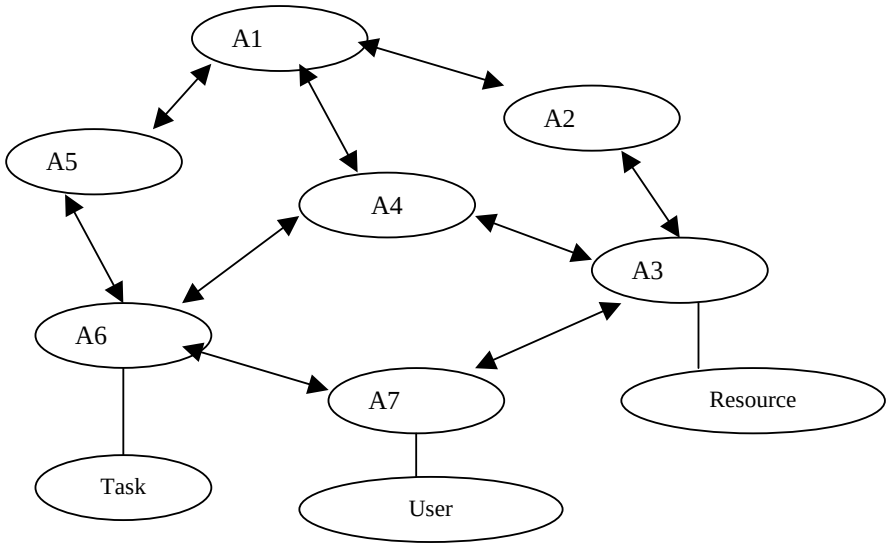


Figure 3.2 Agent Based system architecture

Agent-based Resource Management Architecture is illustrated in Figure 3.2. The main components include Grid Users, Grid Resources and Agents. A Grid resource can be considered a service provider of high performance computing capabilities. Agents are the main components in the system. Agents are organized into a graph. The graph of homogenous agents provides a meta-level view of the Grid Resources. The service

information of each Grid resource can be advertised in the agent graph; agents can also cooperate with each other to discover available resources.

3.4.1 Agent based Resource Discovery process

Each agent has AKBs (Agent Knowledge Base) maintained by resource advertisement. An agent takes the contents in AKBs as its own knowledge, which is mainly used for resource discovery. A resource discovery process is triggered by the arrival of a task in an agent. A resource is usually composed of several parts:

(a) Resource information: These include details of idle resources. This information may be combined with information in AKBs to produce high-level performance information of corresponding resources.

(b) Resource performance: This includes details of resource performance information, which may be used for matchmaking for agents to make decisions on whether a resource can provide a capable service or not.

(c) Options: Additional options may be attached to each resource, which may include user control information for the resource discovery. For example, the user may limit the time and scope of a discovery process.

An agent can query AKB on a resource in a number of ways, for instance:

- Agent searches its AKB and if Agent got request resource information gives back to requesting agent, so the discovery ends successfully.
- Agent searches its AKB but could not find information of requested resource then it can transfer the task to other agents for further discovery.
- Agent searches its AKB but could not find information of requested resource and Agent has no information of other Agents that can process request. thus discovery has failed.

3.4.2 A4 Methodology

Agent based Resource Discovery uses The A4 (Agile Architecture and Autonomous Agents) methodology [28], which is used to build large-scale distributed software systems that exhibit highly dynamic behavior. It is intended that an entire system be built of a hierarchy of identical agents with the same functionality. Agents are considered both as service providers and as service requestors [11]. The implementation of system functions is abstracted to the processes of service advertisement and service discovery.

The A4 model comprises a hierarchical model, a discovery model and a coordination model. The hierarchical model describes a simple method that can organize large numbers of agents. The discovery model solves the problem of coordinating the services of different agents, so that they locate each other, this being the basis of the hierarchical model. The coordination model focuses on the way that services can be organized to provide more complex services.

The A4 methodology considers the following issues:

- While many agents can be utilized as a large-scale multi-agent system, the agents are not predetermined to work together. They possess their own motivation, resource and environment.
- Autonomy is used to describe the character of an agent, with regard to its intelligence and social ability. An agent can fulfill high-level tasks directly or through cooperation with other agents.
- Architecture is used to provide a framework for interaction between agents. For example, multiple agents can be organized into a hierarchy.
- Agility is used to describe the character of the architecture, and implies quick adaptation to environmental change. While autonomy provides the system with component-level adaptability, agility provides the architecture-level adaptability of the system.

Performance metrics of the Agent based Systems

There are four main performance metrics of the agent system: Resource Discovery speed, system efficiency (the balance between resource advertisement and discovery), load balancing and discovery success rate. These Resource Discovery metrics are often conflictive, in that not all metrics can be high at the same time. For example, a quick discovery speed does not equate to high efficiency, as it may be achieved through the high workload encountered in resource advertisement and data maintenance, leading to low system efficiency.

3.5 Query-based Resource Discovery

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. There are a number of query-based approaches as described below.

3.5.1 Directory Services

A directory service is a service that provides read-optimized access to general data about entities, such as people, corporations, and computers via a network protocol. Often, directory services [29, 30] include mechanisms for replication and data distribution.

One approach to constructing a Grid information service is to push all information into a directory service. While this approach pioneered information services [31] for the Grid, the strategy of collecting all information into a database inevitably limited scalability and reliability. X.500, LDAP and UDDI are some of the most popular directory service standards.

(a) X.500

The X.500 standard defines a directory service [32] that can be used to provide extensible distributed directory services within a wide area environment. X.500 provides a framework that could, in principle, be used to organize the information that is of interest to us. However, it is complex and requires ISO protocols and the heavyweight ASN.1 encodings of data. For these reasons, it is not widely used.

(b) LDAP

The Lightweight Directory Access Protocol (LDAP) [30] is a simplified version of the X.500 Directory Access Protocol, which specifies a means of organizing, and accessing information directories over the Internet. It removes the requirement for an ISO protocol stack, defining a standard wire protocol based on the IP protocol suite. It also simplifies the data encoding and command set of X.500 and defines a standard API for directory access.

LDAP is often used in organizations as a means of storing personnel, service, and network topology information. Users of LDAP access information organized in a hierarchical directory tree. Each level of the tree contains attribute-value pairs of information as well as links to lower levels. While LDAP by itself is not a candidate for the role of Resource Discovery solutions in OGSA Grids (it is a means of storage and

organization, not of description and discovery), LDAP's flexibility has made it the choice for a number of Resource Discovery solutions, including MDS the Monitoring and Discovery System used by the Globus toolkit [33].

The **Globus Toolkit's Monitoring and Discovery System (MDS)** [33] uses LDAP to publish information about the current state of resources in a Grid environment. Information is structured as a set of entries, where each entry comprises zero or more attribute-value pairs. The type of an entry, called its object class, specifies mandatory and optional attributes. An Index Service provides the capabilities of the OGSA Information Service and includes data refreshing mechanisms that prevent stale data from being returned in query results. MDS's Trigger Service monitors MDS's data catalog for certain preset conditions, providing a means for asynchronous alarms and warnings to be sent to interested parties.

(c) UDDI

UDDI [34] is an OASIS standard protocol that defines a "standard method of publishing and discovering network-based software components in a Service Oriented Architecture". It provides its functionality through four principle entities: the businessEntity, businessService, tModel, and the bindingTemplate.

The businessEntity is the largest container within UDDI. It contains information about any organizational unit which publishes services for consumption.

In the Grid context, businessEntities can be used to hierarchically separate and form relationships between different organizational groups within a Grid or multiple Grids.

Each service that a businessEntity offers is described by a businessService object. These objects provide information about the service name, categorization, and any other useful details added in a variety of attribute lists. The information is purely descriptive and does not include any instructions for accessing or using the service.

The bindingTemplate object represents the link between abstract businessService descriptions and actual endpoints at which these services may be accessed. Like all objects in UDDI, bindingTemplates are given universally unique identifiers, known as UUIDs, which are used as a key of reference. Each businessService object stores the UUID keys of bindingTemplates within the same businessEntity that provide instances of that service.

BindingTemplates may provide specialized information about a particular businessService endpoint through use of tModels. The tModel is the standard way to describe specific details about a particular businessService or bindingTemplate in UDDI. tModels contain lists of key-value pairs used for description and may be associated with multiple objects in the UDDI database. They may also contain placeholders for URIs which point to descriptive information external to the UDDI registry.

However, to date, Web service-based Grids, such as those based on the emerging Open Grid Services Architecture (OGSA), have not utilized UDDI as a discovery mechanism due to the following reasons:

- It lacks a rich query model due to its lack of explicit data typing and its inability to easily perform bindingTemplate queries based on the values contained within associated tModels.
- It is not well equipped to handle environments that contain resource providers with unpredictable availability because of its limited support for the expiration of stale data.

3.5.2 Peer to Peer Approach

Ian Foster, Adriana Iamnitchi and Daniel C. Nurmi introduced the concept of peer to peer approach [24, 35] for Grid service discovery. According to them, a Resource Discovery solution should consist of the following four components:

- **Membership protocol**: The membership protocol specifies how new nodes join the network and how nodes learn about each other.
- **Overlay construction function**: The overlay construction function selects the set of collaborators from the local membership list. This set may be limited by such factors as available bandwidth, message-processing load, security or administrative policies, and topology specifications.
- **Preprocessing**: Preprocessing refers to off-line processing used to enhance search performance prior to executing requests. An example of a preprocessing technique is dissemination of resource descriptions that is, advertising descriptions of the local resources to other areas of the network for better search performance and reliability.

- **Request processing**: The request-processing function has a local and a remote component described below.
  - *Local processing*: The local component looks up a request in the local information, processes aggregated requests (e.g., a request for A and B could be broken into two distinct requests to be treated separately), and/or applies local policies, such as dropping requests unacceptable for the local administration.
  - *Remote processing*: The remote component implements the request propagation rule i.e., sending the requests to other peers through various mechanisms such as flooding, forwarding, epidemic communication etc.

3.5.3 Parameter Based Approach

Muthucumaru Maheswaran and Klaus Krauter introduced a concept called the "Grid potential" [36] that encapsulates the relative processing capabilities of the different machines and networks that constitute the Grid.

The Grid potential concept is similar to the time-to-live idea used in the Internet. Informally, the Grid potential at a point in the Grid can be considered as the computing power that can be delivered to an application at that point on the Grid. The computing power that can be delivered to an application depends on the machines that are present in the vicinity and the networks that are used to interconnect them. Consequently, a high-performance machine when connected to the Grid will induce a large Grid potential. This potential, however, will decay as the launch point of the application moves away from the point at which the machine is connected to the Grid. The rate of potential decay depends on the network link capacities.

A node in the Grid has several attributes that can be categorized as rate-based attributes and non rate-based attributes. Examples of rate-based attributes include CPU speed, FLOP rating, sustained memory access rate, and sustained disk access rate. The Grid potential is based on the computing power or operating rate of a node. Therefore, to characterize a node for deriving the Grid potential only rate-based attributes are considered. A node (Resource/Service) in the Grid can then be characterized by a vector where each element of the vector is an attribute-value pair and discovery can be done using this vector.

3.5.4 QoS Based Approach

Yun Huang and Nalini Venkatasubramanian [37] addressed the problem of Resource Discovery in a Grid based multimedia environment, where the resources providers, i.e. servers, are not available all the time but are only intermittently available.

Multimedia applications are resource intensive, and consume significant network, storage and computational resources. Furthermore, multimedia applications also have Quality-of-Service (QoS) requirements that specify the extent to which properties such as timeliness can be violated. Quality of Service (QoS) [38] is the summarizing term for the main non-functional properties of such systems that mainly includes reliability, security, performance and adaptability.

3.4.5 ClassAd Matchmaking

Matchmaking is a distributed resource management mechanism developed as part of the Condor project for Grid systems. The matchmaking is based on the idea that resources providing services and clients requesting service advertise their characteristics and requirements using classified advertisements (classads) [39,40]. A matchmaker service that may be either centralized or distributed matches the client requests to the appropriate resources. The matchmaking framework includes several components of a Resource Discovery mechanism (Figure 3.3).

The classad specification defines the syntax and semantic rules for specifying the evaluating the attributes associated with the characteristics and requirements. The advertising protocol lays down the rules for disseminating the advertisements. The following figure shows the actions involved in the matchmaking process.

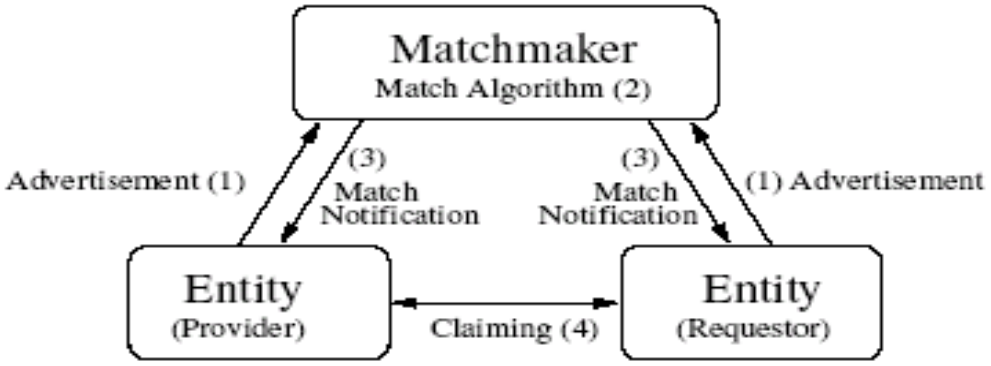


Figure 3.3 Actions involved in Matchmaking process [39]

3.5.6 Ontology based Semantic Matching

An Ontology defines a common vocabulary that facilitates the sharing of information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and their relations. In the Grid environment (GE) where so many different implementations are available, the need for semantic matching based on a defined Ontology becomes increasingly important. The loose coupling between resource and request descriptions removes the tight coordination requirement between resource providers and consumers.

A. Harth, H. Tangmunarunkit ,C.Kesselman and S. decker have designed and prototyped a matchmaker [41] using TRIPLE to use ontologies encoded in W3C's Resource Description Format (RDF) and rules (based on Horn logic and FLogic) for resource matching (Figure 3.4). Resource descriptions, request descriptions, and usage policies are all independently modeled and syntactically and semantically described using RDF schema. Finally, inference rules are used for reasoning about the characteristics of a request, available resources, and usage policies to appropriately find a resource that satisfies the request requirements.

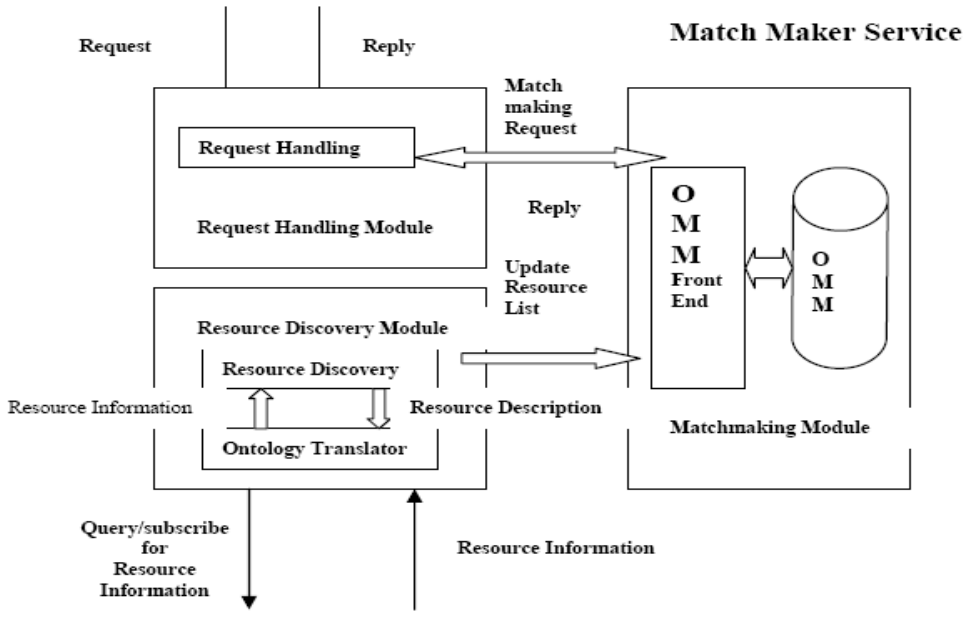


Figure 3.4 Architecture of the Ontology-based Resource Matchmaker Service [41]

3.6 Survey on Resource Discovery Approaches in Existing Grid Systems

Survey is based on an evaluation framework that contains a set of evaluation criteria that are discussed below [42]

• **Query Language and Advertising Language**

Language support for query formulation as well as service and resource description is crucial for the discovery process. It is important to evaluate how expressive is the language and how easy it is to formulate queries and advertisements which are used during the discovery process. Special attention should be given to the semantic support that the language provides to express different aspects of advertisements and queries.

• **Scalability**

Scalability is an important issue when talking about systems that have to accommodate changes in users, resources, etc. It is important to analyze how a system reacts when there is grow in one or more of its dimensions. Another aspect that must be considered is how the scalability of the subsystems, or related systems, affects the scalability of the whole system. The scalability of a discovery mechanism is for example heavily influenced by the scalability of the sub-components such as: query processor, storage, etc.

• **Reasoning support**

The provision of an automatic discovery mechanism can be significantly improved if machine processable descriptions are provided. These descriptions are further checked if they matched against each other (request versus services). New knowledge can be inferred based on existing facts (domain ontologies and domain background knowledge) and this knowledge can be further used during the discovery process. Therefore it is important to identify the reasoning support provided to enable the discovery process. Of course, this criteria is very closely related to Query Language and Advertising Language mentioned above.

• **Matchmaking vs. Brokering**

When talking about the interactions between parties during the process of discovery two approaches can be distinguished:

- *Matchmaking*

  The entity which performs the matching determines if the request and the advertisements match but does not interfere in the next step which is the interaction between matched request and advertisements.

- *Brokering*

  The entity, which performs the matching, determines if the request and the advertisements match and furthermore controls the interaction between matched request and advertisements. Many of Grid approaches for discovery distinguish between these two facets of the discovery process. Therefore it is important to evaluate an approach based on the support that it provides for pure matchmaking or brokering or both.

On the basis of above point a framework can be formulated to evaluate some of the most prominent approaches in Grid environments with respect to Resource Discovery.

(a) Globus

Globus [17] offers Grid information services via an LDAP-based network directory called Metacomputing Directory Services (MDS) [33]. Initial versions of MDS (version lower than 2.4) are based on LDAP network directory. Later versions of MDS (version higher than 3) and later releases are OGSA based. The MDS3 component of the Globus Toolkit Version 3.2 provides information about Grid resources for use in Resource Discovery, selection, and optimization. The MDS3 component is therefore a broad framework that includes any part of GT3 that generates, registers, indexes, aggregates, subscribes, monitors, queries, or displays Service Data in some way. The Index Service combines ServiceDataProviderExecution components with Data Aggregation and ServiceGroup components to create a dynamic data generating and indexing node, similar in concept to a MDS2 hierarchical GIIS. Index Services can be combined in a variety of topologies, useful in building Virtual Organizations. MDS follows both a push and pull protocols for resource dissemination. Higher-level tools such as resource brokers can perform Resource Discovery by querying MDS using LDAP and XML query protocols. Customers describe required resources through a resource specification language (RSL) that is based on a pre-defined schema of the resources database. A resource co-allocator

performs the task of mapping specifications to actual resources, which is responsible for coordinating the allocation and management of resources at multiple sites. RSL allows customers to provide very sophisticated requirements, but there is no possibility for resource providers to specify their constraints on customers. The MDS namespace is organized hierarchically in the form of a tree structure [24]. However this hierarchical model does not provide sufficient performance and expressiveness.

(b) Condor

Due to the lack of expressive and efficient matchmaking in Grid environments Condor was used. Condor [43,44], that is used for high-throughput, computing is a matchmaking framework, which was developed with classified advertisement (ClassAd) for solving resource allocation problems in a distributed environment with decentralized ownership of resources. This framework provides a bilateral match where both resource providers and consumers specify their matching constraints, e.g. policy and requirements. A symmetric requirement is then evaluated for each request-resource pair to determine whether there is a match or not [45].

The five basic components of the matching framework are:

- ClassAd specification
- Advertising protocol
- Matchmaking algorithm
- Matchmaking protocol
- Claiming protocol

(c) Peer-to-Peer Approach

Similar to Grids, P2P [24] systems are build on the principles of cooperation and sharing of resources. P2P systems such as Gnutella, KaZaA, and eMule have proven their applicability for global-scale Resource Discovery and file sharing. Recent contributions to name-based resource location solutions have been proposed in the context of P2P file-sharing systems, such as Gnutella and Napster. The basic mechanism used in Gnutella is flooding. Its flooding-based membership component manages a highly dynamic set of members (with median lifetime per node of about 60 minutes by sending periodic messages. Its overlay function selects a fixed number of nodes from those alive (in most instances, the first nodes of the membership list). Flooding is also the core of the request-

processing component: requests are propagated in the overlay until their time-to-live expires. No preprocessing component is active in Gnutella. Answers are returned along the same trajectory, from node to node, to the node that initiated the request. Gnutella's relatively good search performance (as measured in number of hops) is achieved at the cost of intensive network use.

Another system Napster uses a centralized approach: a file index is maintained at a central location, while real data (files) are widely distributed on nodes. The membership component is centralized: nodes register with (and report their locally stored files to) the central index. Hence, the request-processing component is a simple lookup in the central index. Napster does not use a distinct overlay function.

The Resource Discovery Solution consists of following four components:

- Membership protocol
- Overlay construction function
- Preprocessing
- Request processing

These components have been described in section 3.5.2

A P2P system can use any one of the following request propagation strategies:

- *Random walk*: the node to which a request is forwarded is chosen randomly. No extra information is stored on nodes.
- *Learning-based*: nodes learn from experience by recording the requests answered by other nodes. A request is forwarded to the peer that answered similar requests previously. If no relevant experience exists, the request is forwarded to a randomly chosen node.
- *Best-neighbor*: the number of answers received from each peer is recorded (without recording the type of request answered). A request is forwarded to the peer who answered the largest number of requests.
- *Learning-based + best-neighbor*: this strategy is identical with the learning-based strategy except that, when no relevant experience exists, the request is forwarded to the best neighbor.

(d) Semantic Approach

Although traditional service discovery methods may be effective when *a priori* knowledge of the services or agreements about implicitly shared ontologies can be assumed, they fail to scale to large, dynamic, open, environments, where a high degree of autonomy is required [46]. Service discovery in Grid environments to date are only based on particular keyword queries from the user. One of the problems with keyword-based service discovery approaches is that they cannot completely capture the semantics of a user's query because they do not consider the relations between the keywords. One possible solution for this problem is to use retrieval based on semantics [46].

Semantic web service discovery overcomes this limitation by providing an ontological framework by which services may be described and processed. Semantic description of implicit community knowledge offers a mechanism to cope with the heterogeneity of resources by providing a rich descriptive framework and common vocabulary to integrate and search over apparently disparate data, services and workflows.

Ontology Based Matchmaker (**OMM**)

When two or more parties seek a common understanding of something, they must work together to ensure that there is a high degree of correlation and similarity between the details of their respective descriptions and definitions of what they are trying to agree on. OMM [45] is an asymmetric description based semantic approach. It provides a flexible and extensible approach for performing Grid resource selection using an Ontology-based matchmaking technique and algorithm. Unlike the traditional Grid resource selectors (like Condor-G) that describe resource and request properties based on symmetric flat attributes, separate ontologies (i.e., semantic descriptions of domain models) are created to declaratively describe resources and job requests using an expressive Ontology language. Instead of exact syntax matching, Ontology-based matchmaker performs semantic matching using terms defined in those ontologies. The loose coupling between resource and request descriptions removes the tight coordination requirement between resource providers and consumers. Matching between request specifications to resource capabilities is done in terms of rules. Similar to other matching systems, matchmaker provides the ability to describe properties and matching preference. Matchmaker also

supports bi-lateral matching and gang matching. The Ontology-based matchmaker consists of three main components:

- *Ontologies* that capture the domain model and vocabulary for expressing resource advertisements and job requests,
- *Domain background knowledge*, capturing additional knowledge about the domain,
- *Matchmaking rules*, defining when a resource matches a job description.

Minimizing false positives and false negatives is achieved with three selection stages in combination with well-defined ontologies. The selection stages are:

- Context selection, where the request is matched within the appropriate application context.
- Semantic selection, where the request is matched semantically.
- Registry selection, where a lookup is performed.

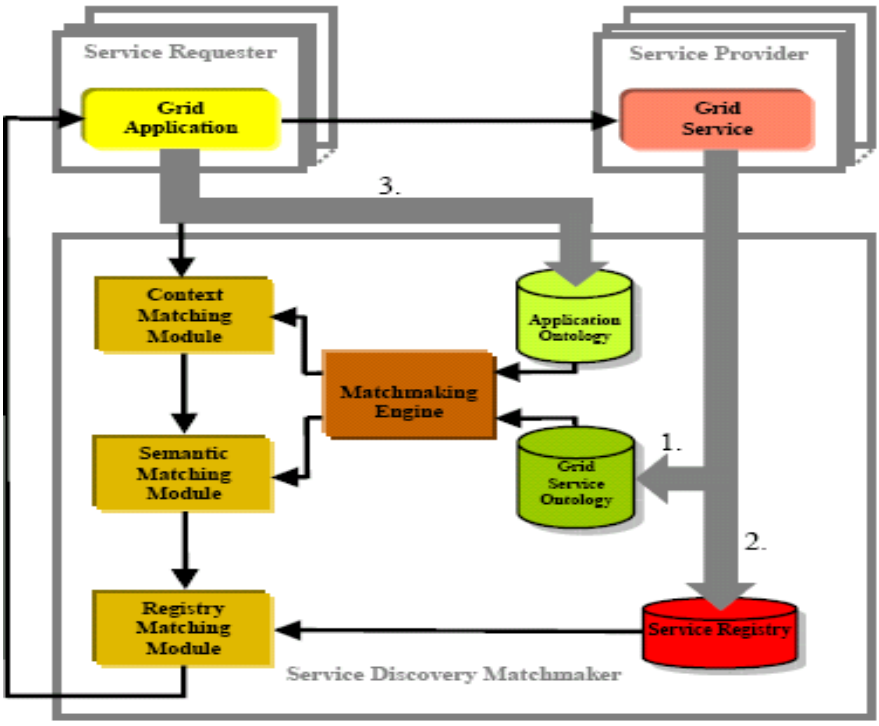


Figure 3.5. Semantic Service Discovery Matchmaker – Registration [45]

Figure 3.5 shows the interactions of a service registration process.First, the service providers need to register their services for the matchmaking process. The service provider registers its service semantics in the Grid service Ontology (1) and the necessary contact details in the service registry (2). Service semantics comprises of a service name, a service description, service attributes (input/output) and metadata information. Furthermore, the service requester specifies the context semantics of the application in the application Ontology (3)

.

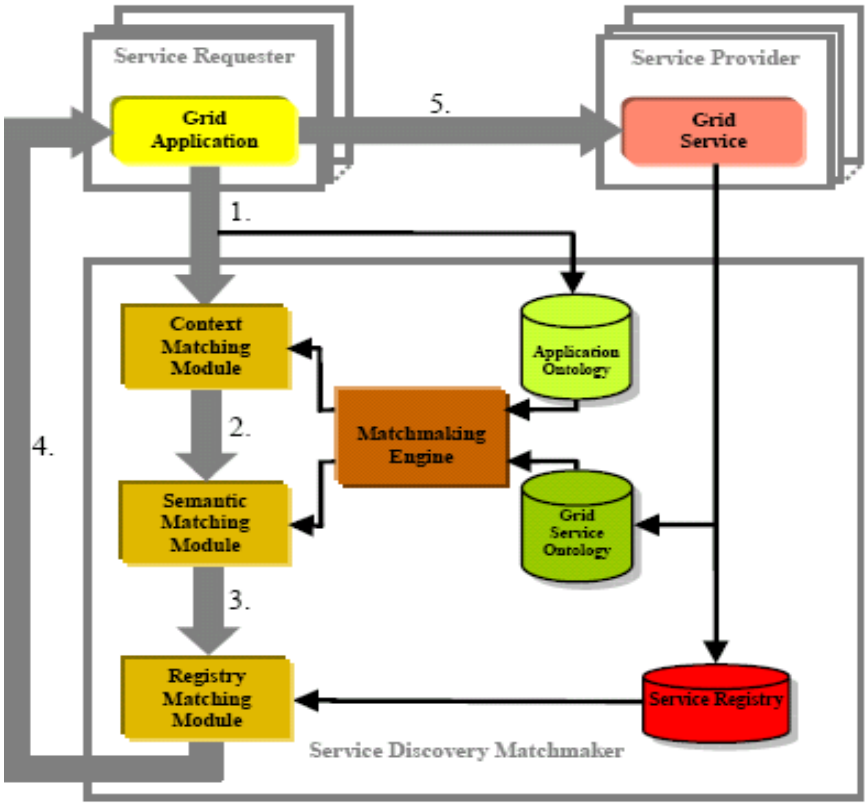


Figure 3.6. Semantic Service Discovery Matchmaker – Matchmaking [45]

The interactions of a service request are shown in Figure 3.6. The Grid application sends out a request to the service discovery matchmaker (1). The request has to go through the context-matching module first. Here, the request is matched within the appropriate context of the application Ontology. This means that depending on the service request, which came from one of the applications, the appropriate context Ontology is chosen and the first match is performed. Additional parameters are attached to the request and

forwarded to the semantic matching module (2). In this module the semantic match is performed. Semantic matchmaking allows the service request to be matched using the semantics (metadata) of services. Having all necessary semantic data, a service lookup is done using a service registry (3). This lookup information is sent back to the Grid - application (4) to be used for the Grid service call (5).

Approach/ Middleware	**Query and Advertising Language**	**Scalability**	**Reasoning support**	**Matchmaking vs. Brokering**
Globus	RSL: Globus Resource Specification Language	Distributed query based service discovery	None	Globus does not provide matchmaking or brokering
Condor	ClassAd (Classified Advertisement) Language	Centralized query based Resource Discovery	Constraint-based reasoning	Matchmaking
P2P Structured Overlay	XQuery	Supported by Distributed Hash Table (DHT)	Keyword searching by distributing inverted indices among hosts by keyword in the DHT.	Brokering
Ontology based Matchmaker	RDF, RDF schema	Centralized querying mechanism	Deductive Database support	Matchmaking and Brokering

Table 3.1 Comparison of existing approaches in Resource Discovery Grid Systems

3.6.1 Comparison of existing approaches in Resource Discovery Grid Systems

Based on above study following Comparison of existing approaches in Resource Discovery Grid Systems has been concluded in Table 3.1.

This chapter described Resource Discovery process and various approaches that can be used for Grid Resource Discovery. Alongwih that a comparison of Grid Resource Discovery approaches used in various Grid systems has also been given in this chapter.
In the next chapter we will focus on the problems encountered in the existing approaches and possible solutions to these problems.

Chapter 4

Problem Formulation

This chapter gives a detailed description of the problem found in existing approaches of Resource discovery in a Grid environment and the semantic solutions that can be used to solve this problem.

4.1 Limitations of Existing Approaches

The description of a resource is essential for automated discovery and search, selection, matching, composition and interoperation, invocation and execution monitoring; and different Middleware specify different rules for describing a resource. Hence the information gathered from these diverse sources tends to be semantically heterogeneous and needs to be correlated. The sources of information are equally different or heterogeneous. None of the approaches described in the previous chapter is capable of correlating this information. Thus there is a need to construct systems that are capable of answering intelligently by understanding the terms involved and the relationships between them.

Existing resource description and resource selection in the Grid is highly constrained [4]. Traditional Resource Discovery is done based upon symmetric attribute-based keyword matching with the support of simple query languages. In these systems, the values of attributes advertised by nodes are compared with those required by jobs. For the comparison to be meaningful and effective, the node providers and consumers have to agree upon attribute names and values beforehand. However, exact keyword matching and coordination between providers and consumers make such system inflexible and difficult to extend to new characteristics or concepts. Moreover, in a heterogeneous multi-institutional environment such as the Grid, it is difficult to enforce the syntax of node descriptions since a Grid is setup using a number of Middleware, all of which describe the resources in their own way. Thus there is a need for a system that is independent of the syntax used for resource descriptions and provides a semantic

approach for finding resources based on their concept similarity rather than exact matching of their attribute value.

Shortcomings of Syntax based Resource Descriptions

As seen in previous chapter Attribute based resource description (used in most of the Grid systems) that is used in match making process of Resource Discovery has several shortcomings that are listed as follows;

- Lack of expressiveness
- Lack of Dynamicity
- Lack of Independence between the Resource Provider and Resource Consumer
- Symmetric Resource Descriptions

In such Systems the mechanism of Resource Description is symmetric, i.e. both resource provider and consumer have to agree on certain syntax or schema of resource description. The exact matching and coordination between providers and consumers make such systems inflexible and difficult to extend to new characteristics or concepts. Also, in such heterogeneous multi-institutional environment, it is difficult to enforce the syntax and semantics of resource descriptions.

Thus it is necessary to explicitly, precisely and unambiguously describe Grid resources and specify various constraints over resource descriptions, as well as the description should be automatically interpretable and understandable by Grid components.

Shortcomings of syntax based resource discovery can be solved using Ontology because it can provide unambiguous, asymmetric resource description that can be easily shared across various Virtual Organizations. The expressiveness of Ontology makes matchmaking process more efficient by providing list of resources based on semantics as the result of Resource Discovery process. Ontology based Resource Discovery also helps to improve Quality of Service (QoS) of Grid System and provide transparent access of Grid resources that is major goal of any Grid system.

This thesis aims is to design and develop a Resource Discovery Approach based on Ontology, which overcomes the shortcomings of the current state of the art in the context. Main objective of thesis is

- Design an Ontology of Grid Resources to provide unambiguous resource description.

- A Resource Discovery Approach based on Ontology is proposed and implemented to discover resources in distributed environment.

This chapter described the limitations found in existing Resource Discovery approaches and possible semantic solutions to this problem.
The next chapter describes the proposed solution and implementation details of Ontology based Resource Discovery Approach to find resources offering capabilities similar to the one desired by the user.

Chapter 5

Proposed Ontology based Resource Discovery Approach

This chapter describes the proposed Resource Discovery approach design of TU Grid Ontology and experimental setup.

5.1 Ontology based Resource Discovery Approach

Traditional Resource Discovery methods strive for symmetric syntactic description of resource and request properties, so it is difficult to introduce new concepts or characteristics into the system. Moreover, in the Grid environment, where resources and users span multiple organizations, it may be difficult to guarantee that resources and requests will use the same attribute names, and both resource providers and consumers interpret semantics of the same attributes in the same way. Along with this, in Grid environment different resources can provide similar capabilities but with different Quality of Services. The resource capabilities are required to be presented in. such a way that a consumer can easily discover a resource or a resource ensemble with needed capabilities.

Solution to the above stated problems can be provided by Ontology. Ontology provides unambiguous, asymmetric Resource Description that can be easily shared across various Virtual Organizations. The expressiveness of Ontology makes matchmaking process more efficient by providing list of resources based on semantics as the result of Resource Discovery process.

Ontology can be defined as a specification of a conceptualization. In the context of Grid environment, Ontology captures the domain model and vocabulary for expressing resource advertisements and job requests [4]. Resource request and resource description are on the basis of Syntax or Semantics regardless of query or agent based matching of resource description. A comparison between Syntax and Semantic based Resource Discovery has been given in Table 5.1.

Criteria	Syntax based Resource Discovery	Semantic based Resource Discover using Ontology
Description of resource and request	Symmetric	Asymmetric
Sharing and Maintainability of Ontology	Difficult to maintain and share	Easier to maintain and share
Ability to describe matching preference	Very much limited due to matching based on syntax	Matching preference and ranking criteria easily be added with resource and request
Integrity Checking	No integrity checking due to lack of semantics	Integrity checking done on the basis of domain knowledge
Expressiveness	Less expressive	Much expressive due to the asymmetric description, the request can be modeled specifically for domain specific applications
Flexibility and Extensibility	Less flexible and extensible because both resource requester and provider have to agree prior to operation	New concepts and constraints can be easily added into the Ontology at any time

Table 5.1 Comparison between Syntax and Semantic based Resource Discovery

Lets consider a Grid information system scenario where Ontologies are used to describe resources and refer to their metadata to allow resource access. Reference architecture for such Grid information system is depicted in Figure 5.1. In particular there would be following layers:

- **Resource Repository:** This layer contains concrete Grid resources, such as data sources and software components. Such layer contains both metadata of resources, describing details of each concrete resource on each Grid node, and Ontologies that describe semantic properties of resources and semantic relations.
- **Resource Access:** This layer allows general primitives to access heterogeneous resources i.e. the same functions are used to access a text document or a relational

database. The Resource access can be directly implemented by applications using metadata.

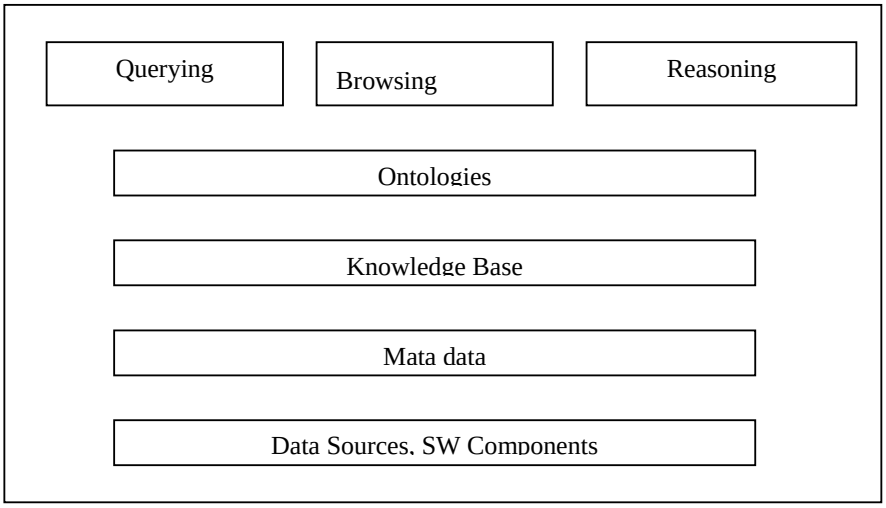


Figure 5.1 Layers in a typical Grid System

- **Information Services**: This layer offers a set of functions to query and browse resources, a approach to describe resources and to publish such information in both metadata and ontologies, and finally functions to infer new knowledge by reasoning or mining.

5.2 Proposed Ontology based Resource Discovery Approach

Ontology based Resource Discovery Approach resides on the top of Middleware (Figure 5.2). It can be embedded in Resource broker that assigns jobs to various interoperable Middleware. Resource Discovery Approach (Figure 5.3) consists of Information services of Grid Systems, Information Collector, Ontology Repository, User Interface, and Matchmaker. Information Services e.g. Condor_status [50] in Condor, provides Resource Information to Information Collector periodically. Information Collector converts this information into standard format and this Resource Information is stored in Ontology Repository. User interacts with System through User Interface.

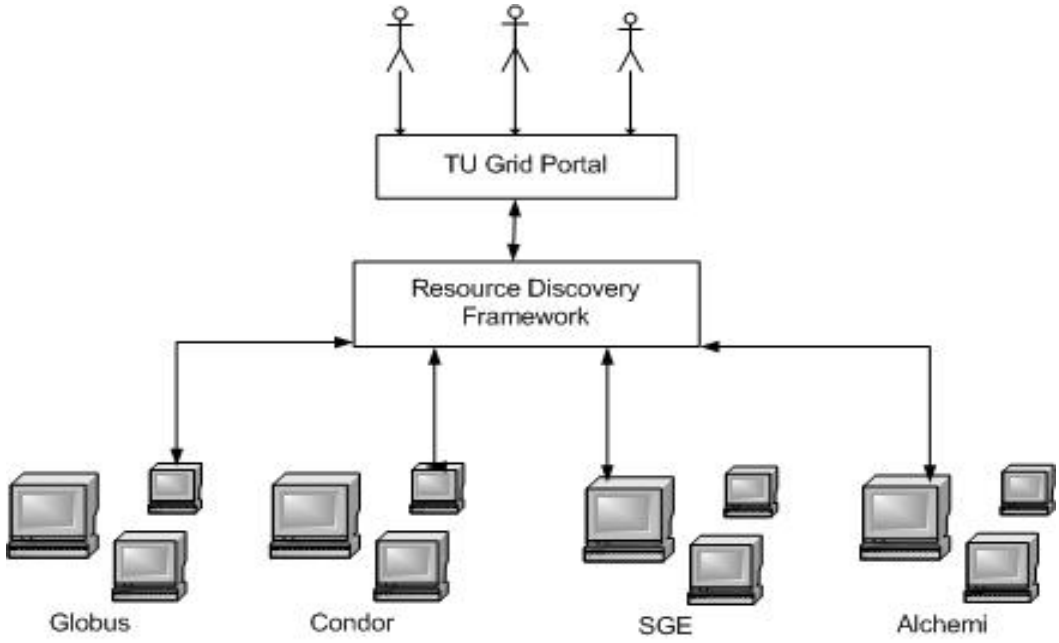


Figure 5.2 Interoperable Grid System Architecture

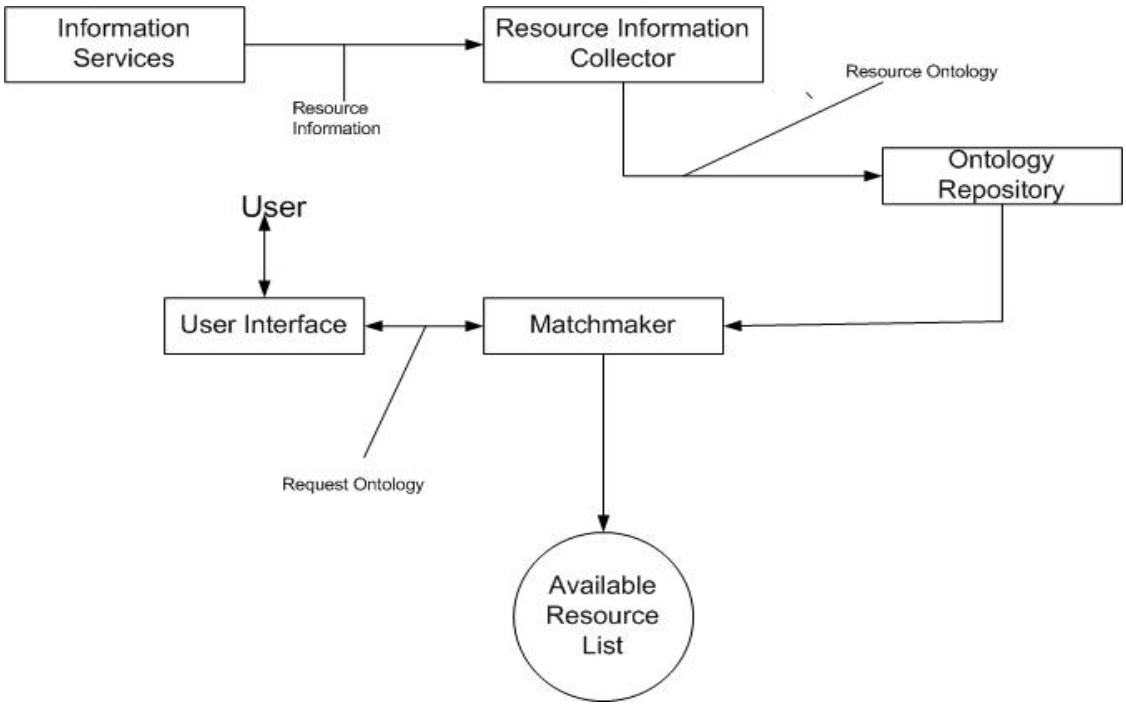


Figure 5.3 Ontology based Grid Resource Discovery Approach

Resource Discovery process starts with submission of user Requirements. User enters CPU Architecture CPU Availability, memory requirement and Operating System of

target machine. Query is formulated on the basis of above information and is send to central manager. Central manager receives the request and find out classes of instances provided in user query e.g. Pentium 4 is instance of Intel Class. The next step is to find out all the equivalent classes of given class if any. In this step all respective instance class and Equivalent class are retrieved, the next step is to find all the instances of given class and equivalent class. Result this step provides a list of instances against a single instance. This list of instance is called as Inferred instance. Now list of instances are searched in dynamic Ontology Repository that is updated in every 10 seconds by Information Collector.

5.2.1 Design of TU Grid Ontology

The TU Grid Ontology is designed to represent Grid knowledge of various Grid systems. Therefore, it should be open and extensible as there are thousands of Grid entities, services, components, and applications of different Grid Middlewares. To deal with the openness and extensibility requirements, the Web Ontology Language (OWL) [51] has been used to describe the terms and classes in the TU Grid Ontology. OWL is a semantic markup language for publishing and sharing Ontologies on the World Wide Web, which is developed as a vocabulary extension of the Resource Description Framework (RDF) [52]. Given the fact that both RDF and OWL are developed under the vision of semantic web and W3C recommendations, the TU Grid Ontology is thus open, and compatible with other systems. Furthermore, the characteristics of the RDF data model make the Ontology easier to extend by either adding new classes in the Ontology or adding extra properties into a defined class without any conflict with existing definitions. The RDF data model is a directed graph with labeled nodes and arcs; the arcs are directed from one node (i.e. subject) to another node (i.e. object). The object may be linked to other nodes (e.g. other new classes) through a property.

One key feature of this data model is that properties in RDF are defined globally, that is, they are not encapsulated as attributes in class definitions. It is thus possible to define new properties that apply to an existing class without changing that class. One main challenge in developing TU Grid Ontology is to provide formal definitions and axioms that constrain the interpretation of classes and terms because till today it is a manual

process. There is no such automation tool that generates Ontology of given domain automatically.

In the following sections, concepts are described and, in particular, represent their constraints on the Grid domain according to the knowledge derived from analyzing, evaluating, and experimenting with different Grid architectures, production Middleware and large Grid infrastructures, such as, Globus, Unicore, Condor, SGE. Firstly, set of TU Grid Ontology classes are defined. Subclasses can then be added accordingly. After that, the relationships and constraints among the Ontology classes using Object and Datatype properties are defined. These properties are links among the defined classes. Finally, the classes, properties, and instances together form a knowledge base that captures the configuration and state of specific Grid infrastructures. Such a knowledge base can be used for various inquiries and for decision-support of end-users, application developers, Grid administrators, etc.

(a) The TU Grid Ontology Classes

In Web Ontology Language (OWL) Classes provide an abstraction mechanism for grouping resources with similar characteristics. Like RDF classes, every OWL class is associated with a set of individuals, called the class extension. The individuals in the class extension are called the instances of the class.

In TU Grid Ontology some core classes have been defined, and its subclasses and description of Top-level classes (Figure 5.4) have been given below:

- **Resource**: This class describes Hardware, Software, computing resource, storage resource and network resource within a Grid. This class is further classified into subclasses. Its subclasses are software that includes classes of operating Syatems, Computingresource that contain cluster, CPU as subclass and data .
- **Middlewares**: This class encapsulates Software that glue Grid services together following a specific Grid architecture. This class is further categorized into subclasses of different Middlewares Condor, Globus, Alchemi, SGE and Unicore.
- **Users:** This class describes types of users who use the Grid system.
- **Applications:** This class encapsulates Applications that can run on Grids.

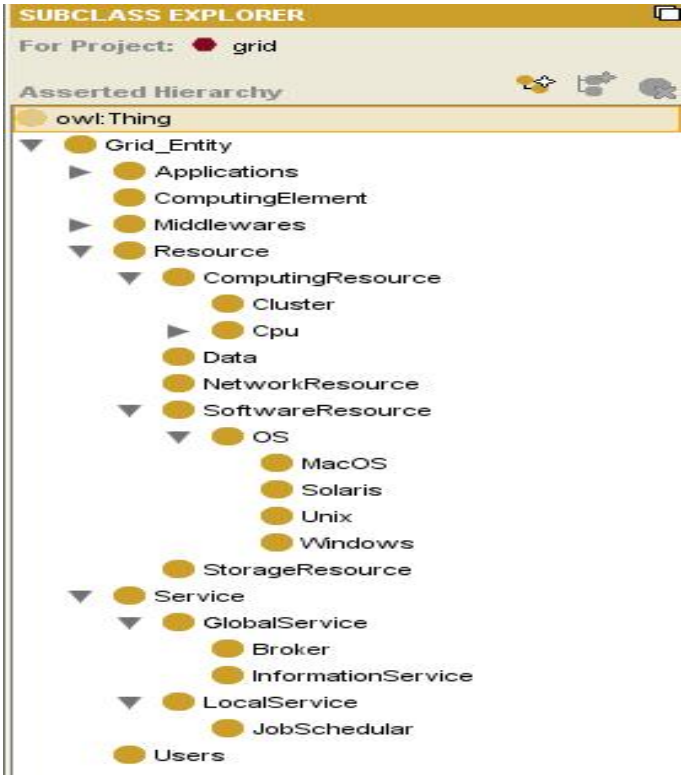


Figure 5.4 TU Grid Ontology Classes in Protégé

- **Service:** This class encapsulates services that can provide one or more Grid functionalities. This class is further categorizes in Localservice class and Globalservice class.Localservice has one subclass Jobshedular and Globalservice have subclasses broker and InformationService.
- **ComputingElement:** This class encapsulates Grid entity and has several kinds of computing resources, such as CPU, Memory etc. The resource is organized by Grid Middleware, user and Grid services to provide computing power to respective domain.

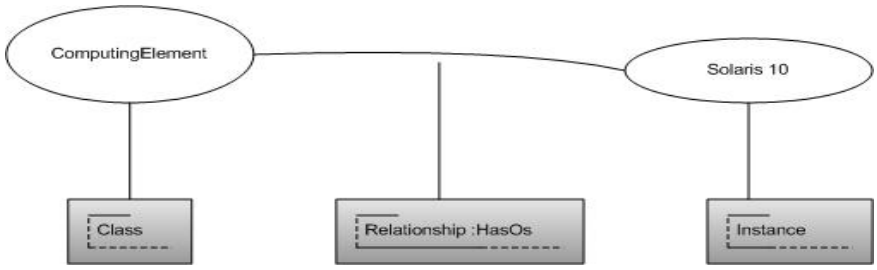


Figure 5.5 Representation of instances and relationship

(b) The TU Grid Ontology Properties

In order to represent the relationships and constrains among the Ontology classes, properties have been defined that provide the semantic meaning for the TU Grid Ontology entities (Figure 5.5).

In TU Grid Ontology two types of properties are defined

- **Object Property:** Object Property is relationship between instances of two Classes.
- **Datatype Property:** Datatype Property is relationship between instances of Classes and RDF literals and XML Schema Datatypes.

Figure 5.6 Object Properties in TU Grid Ontology

Here properties are defiened alongwith domain and range.

Object Properties (Figure 5.6) are as follows:

- **HasOs:** This property links ComputingElement instance to Operating System instance. Domain and range of this property is ComputingElement and OperatingSystem respectively.
- **HasCpu:** This property links ComputingElement instance to CPU instance. Domain and range of this property is ComputingElement and CPU respectively.
- **HasInstalledMiddleware**: This property links ComputingElement instance to Middleware instance. Domain and range of this property is ComputingElement and Middleware respectively.

- **HasService:** This property links ComputingElement instance to Services instance. Domain and range of this property is ComputingElement and Services respectively.

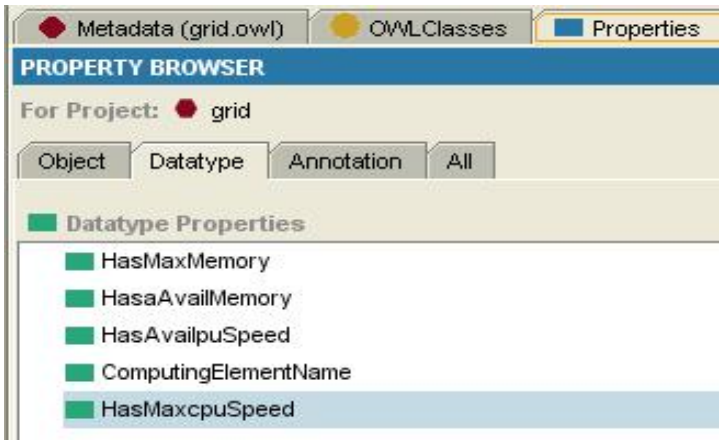


Figure 5.7 Datatype Properties in TU Grid Ontology

Datatype properties (Figure 5.7) are as follows:

- **ComputingElementName:** This property links ComputingElement instance to the name of computing element, which is a XML Datatype. Domain and range of this property is ComputingElement and XML Datatype respectively.
- **HasMaxMemory:** This property links ComputingElement instance to value of maximum primary memory of computing element, which is a XML Datatype. Domain and range of this property is ComputingElement and XML Datatype respectively.
- **HasAvailMemory:** This property links ComputingElement instance to value of availabe primary memory of computing element, which is a XML Datatype. Domain and range of this property is ComputingElement and XML Datatype respectively.
- **HasMaxCpuSpeed:** This property links ComputingElement instance to value of maximum CPU speed of computing element that is a XML Datatype. Domain and range of this property is ComputingElement and XML Datatype respectively.
- **HasAvailcpuSpeed:** This property links ComputingElement instance to value of available CPU speed of computing element, which is a XML Datatype. Domain and range of this property is ComputingElement and XML Datatype respectively.

The classes in TU Grid Ontology (see in Figure 3) are fundamental concepts, elements, and aspects of a Grid system. They provide a framework for representing any entity of a Grid system. In other words, all-important Grid entities and resources in/on Grids can be located within this Approach. For example, to describe a condor-based Grid ComputingElement (CE), which is responsible for providing computing power and is comprised of one or more similar machines managed by a single job scheduler, following aspects need to be determined

- Which GridMiddleware has been installed?
- What kinds of Services run on it?
- Which kinds of applications it can run?
- How many Grid resources it can provide?

Therefore, a condor CE can be represented using the defined TU Ontology classes in the RDF triple model as follows:

- CE isA ComputingElementg;
- CE HasService Serviceg;
- CE HasInstalledMiddleware GridMiddlewareg;
- CE hasMaxCpuSpeed XMLSchema#Float g;
- CE HasAvailCpuSpeed XMLSchema#iFloat g.

5.2.3 Ontology Repository Schema Design

A Repository schema has been designed for TU Grid Ontology. MySQL is used for Repository [53], which is an open source Database. Repository Schema is presented in figure 5.8.

This schema contains following Tables

- Class: This Table contains information of all classes of Ontology, its subclasses and Equivalent classes if any.
- Instance: This Table contains information of all instances of Ontology classes.
- Resource: This Table contains information of computing resource and its relationships as defined in owl file.
- Relationship: This Table contains all the relationships of a Computing Element.

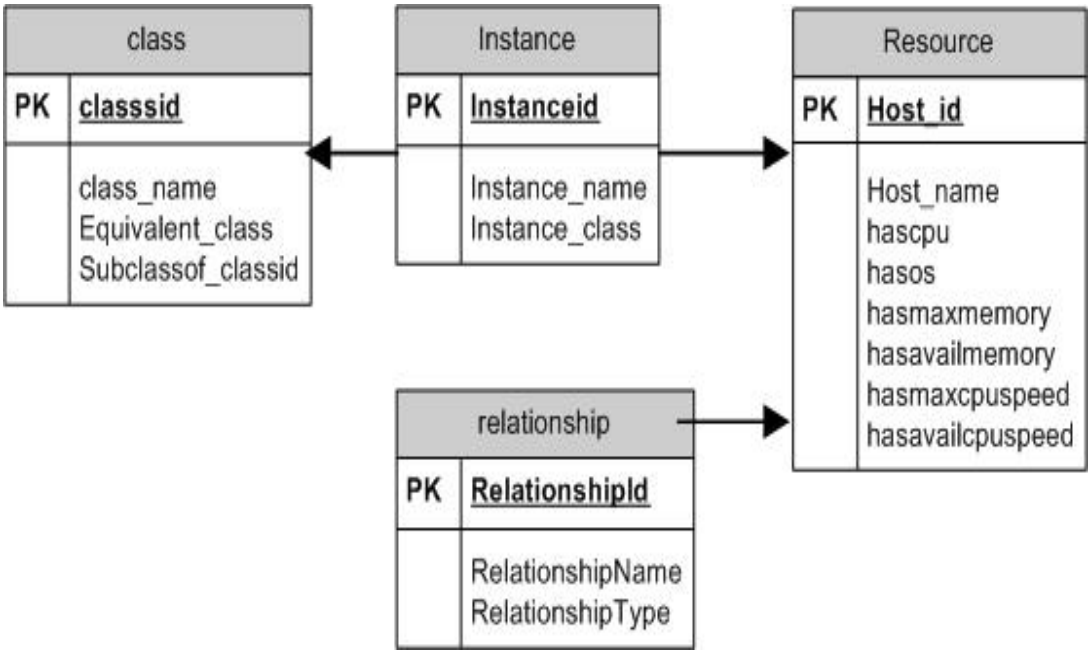


Figure 5.8 Ontology Repository Schema

5.3 Experimental Setup

Throughout the implementation of the Resource Discovery Approach emphasis has been on the use of Open Source Technologies and Toolkits. TU Grid Ontology is designed in protégé [54] that is an open source Ontology Editor .The Ontology based Resource Discovery Approach has been implemented in JAVA programming language. Jena Framework [55] is used for extraction of Ontology classes relationships and instances. Apache Tomcat [56] is used for handling Http request of user. The steps followed for the installation of above discussed tools and framework.

Step 1: Installation of Protégé

Protégé is a free, open source Ontology editor and knowledge-base framework. Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development Protégé is freely available and can be downloaded from

http://www.protégé.stanford.edu/download/download.html

Installing Protégé on Windows can be done easily using the Windows installer. Its interface and functionality is similar to other wizard-based installers.

Step 2: Installation of Jena Framework

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena is open source and grown out of work with the HP Labs Semantic Web Programme.

The Jena Framework includes:

- A RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage

To use Jena in your application, ensure all the .jar files in the lib/ are on the classpath or available to your application.

Step 3: Installation of Apache Tomcat

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.Apache Tomcat is developed in an open and participatory environment and released under the Apache Software License. Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. These are some of the key tomcat directories, all relative to $CATALINA_HOME. CATALINA_HOME is the address of the directory where Apache Tomcat is installed.

- /bin - Startup, shutdown, and other scripts. The *.sh files (for Unix systems) are functional duplicates of the *.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.
- /conf - Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container.
- /logs - Log files are here by default.
- /webapps - This is where your webapps go.

Installing Tomcat on Windows can be done easily using the Windows installer. Its interface and functionality is similar to other wizard based installers, with only a few items of interest.

- Installation as a service: Tomcat will be installed as a Windows NT/2k/XP service no matter what setting is selected. Using the checkbox on the component page sets the service as "auto" startup, so that Tomcat is automatically started when Windows starts. For optimal security, the service should be run as a separate user, with reduced permissions
- Java location: The installer will use the registry or the JAVA_HOME environment variable to determine the base path of a J2SE 5 JRE.
- Tray icon: When Tomcat is run as a service, there will be a tray icon visible. Note that when choosing to run Tomcat at the end of installation, the tray icon will be loaded at the same time.

Step 4: Installation of MySQL

MySQL is widely used on the Internet as well as on other applications. The software is available under both General Public License (GPL) and commercial licenses. The former license qualifies MySQL as legitimate open source software; but since development is for the most part done by MySQL employees, it cannot be considered the best example of an open source community.

Installing MySQL on windows is very simple with windows installer. At the time of installing, it asks for password for root, database type and some other information. Windows firewall needs to be configured to support MySQL as MySQL required 3306 port open for TCP communication.

This chapter focused on the Ontology based Resource Discovery; design of TU Grid Ontology, design of Ontology repository schema and lastly experimental setup to implement this approach.

Next chapter focuses on implementation of Ontology based Resource Discovery approach and its experimental results.

Chapter 6

Implementation Details and Experimental Results

This chapter focuses on implementation of proposed Ontology based Resource Discovery Approach and experimental results of this approach.

6.1 Implementation of Resource Discovery Approach

Proposed Resource Discovery Approach uses Resource Ontology that is specification of classes of resources and its relationships that is defined in previous chapter. Ontology Repository that is designed in previous chapter is populated with Ontology classes, relationships and instances of classes using Jena Framework. A User Interface has been developed to interact with user and Information Service that fetch Host information and update repository periodically. Proposed Resource Discovery Approach can be depicted in Information Flow Diagram (Figure 6.1). Implementation details Components have been given in following section.

6.1.1 Implementation of Resource Discovery Approach Components

Proposed Resource Discovery Approach has been developed using Java, SIGAR API's [57], JSP [58] and MySQL Database. Component Diagram of proposed Approach is shown in Figure 6.2 Discussed Approach is divided into three major modules, which are explained below:

(a) Request Handling Component

This module is developed in JSP and HTML(Screenshot 6.1). It uses client server mechanism to handle user requests. Here Apache Tomcat server has been used to process user request. These modules are responsible for taking user request and give it to central manager. Central manager forward this request to match making module. Matchmaking module performs matchmaking and sends result back to the Grid user.

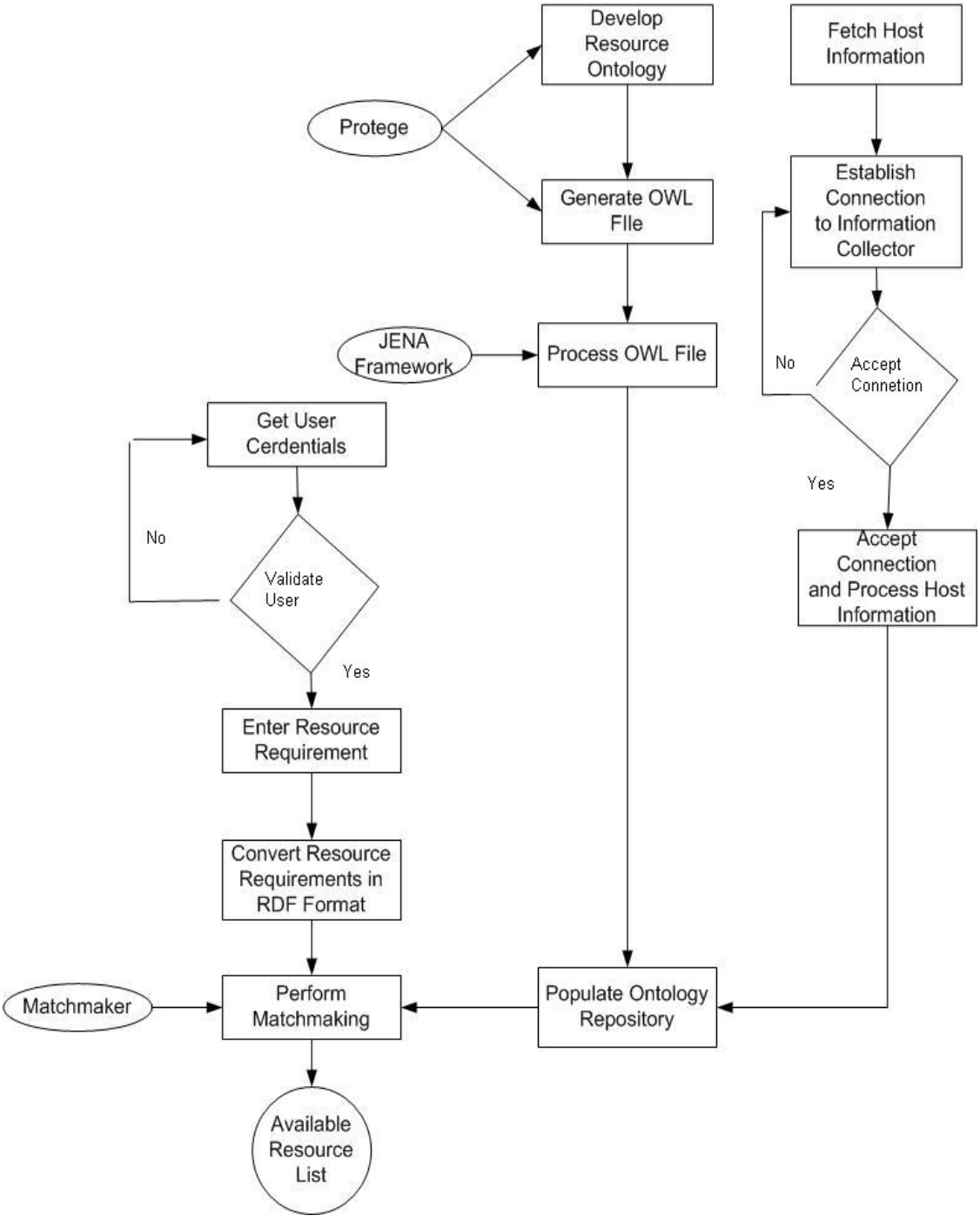


Figure 6.1 Information Flow Diagram of Ontology based Resource Discovery Approach

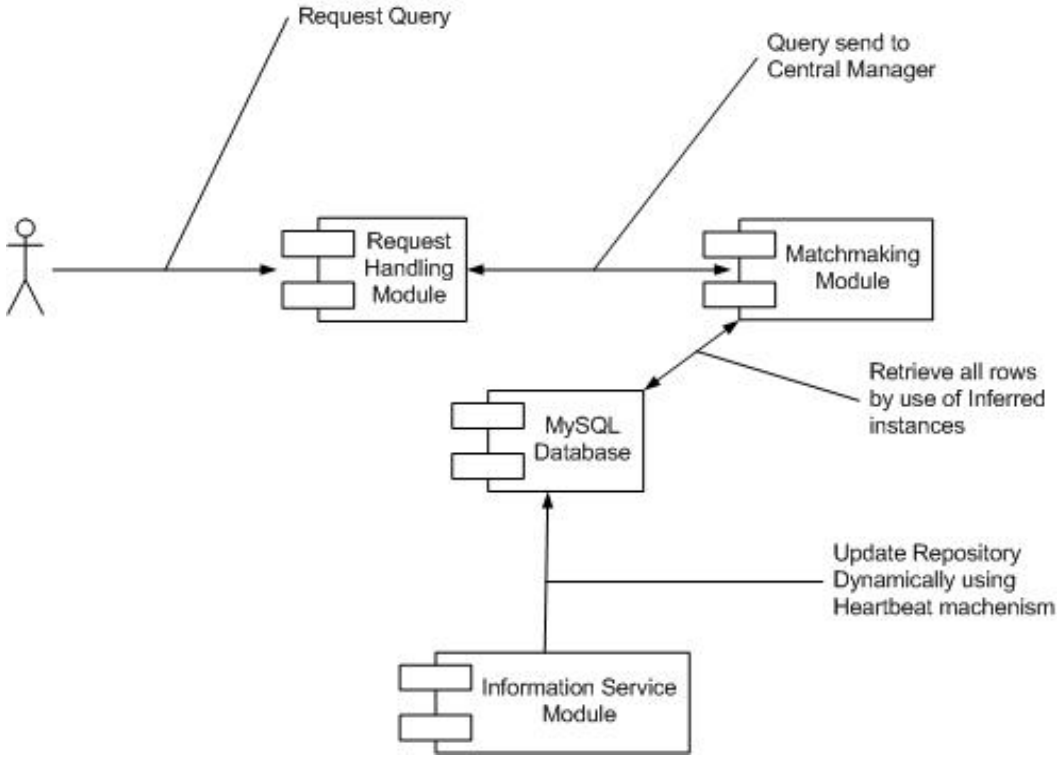
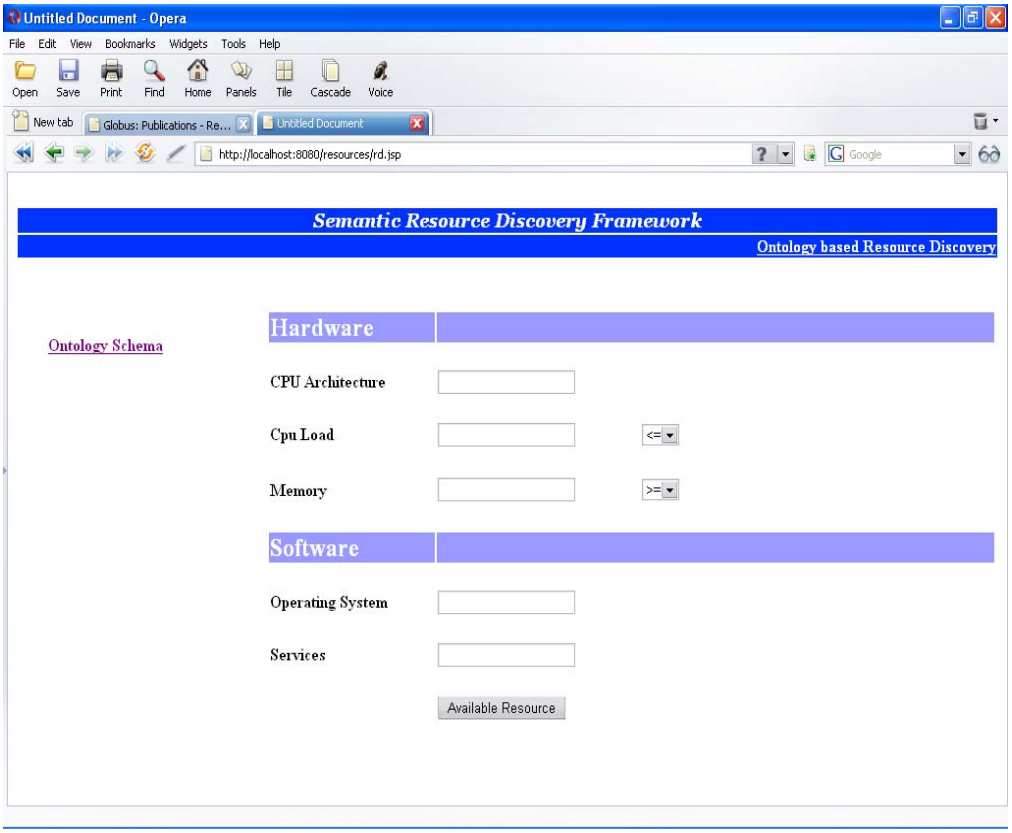


Figure 6.2 Component Diagram of Resource Discovery Approach

(b) Information Service Component

This module is developed using java Sockets, Threads and SIGAR API's. SIGAR (System Information Gatherer and Reporter) is a cross-platform, cross-language library and command-line tool for accessing operating system and hardware level information in Java, Perl and .NET.

As a key part of proposed approach, the Information Service component dynamically collects resource information from multiple sources, it works in two steps. In first step all system information are gathered using SIGAR API's Classes. Then after each client create a socket and connected to server Socket with specified IP and port .In second step At server side, server accept incoming connection and allocate a socket to process client request and all the information are stored in repository. At client side system information repeatedly send to server with a delay of 10 seconds. Whenever client breaks connection to server, server waited for 15 minutes for client connection and if client connection remains closed then server removes the entry of client from repository.



Screenshot 6.1 Use interface of request handling module

(c) Matchmaking Component

Match making module is developed in java and uses MySQL Database as backend. Whenever a user request comes to matchmaking module it creates dynamic Query according to the information provided by the user. Matchmaking performed by retrieving class name of instances. After having class name retrieve equivalent classes if any .now get the list of all instances of specified class as well as equivalent class. These instances are retrieved from repository and result is given back to user.

6.2 Experimental Results

In this section experimental result has been shown using some examples.

Here two Resource Discovery approaches Syntax based resource Discovery and Ontology based Resource Discovery are considered.

User gives CPU Architecture current available CPU Cycles, memory and OS information in request and apache tomcat server accept this request perform matchmaking using

syntax based or Ontology based matchmaking, and give list of available resource list that fulfils user requirement, back to user.

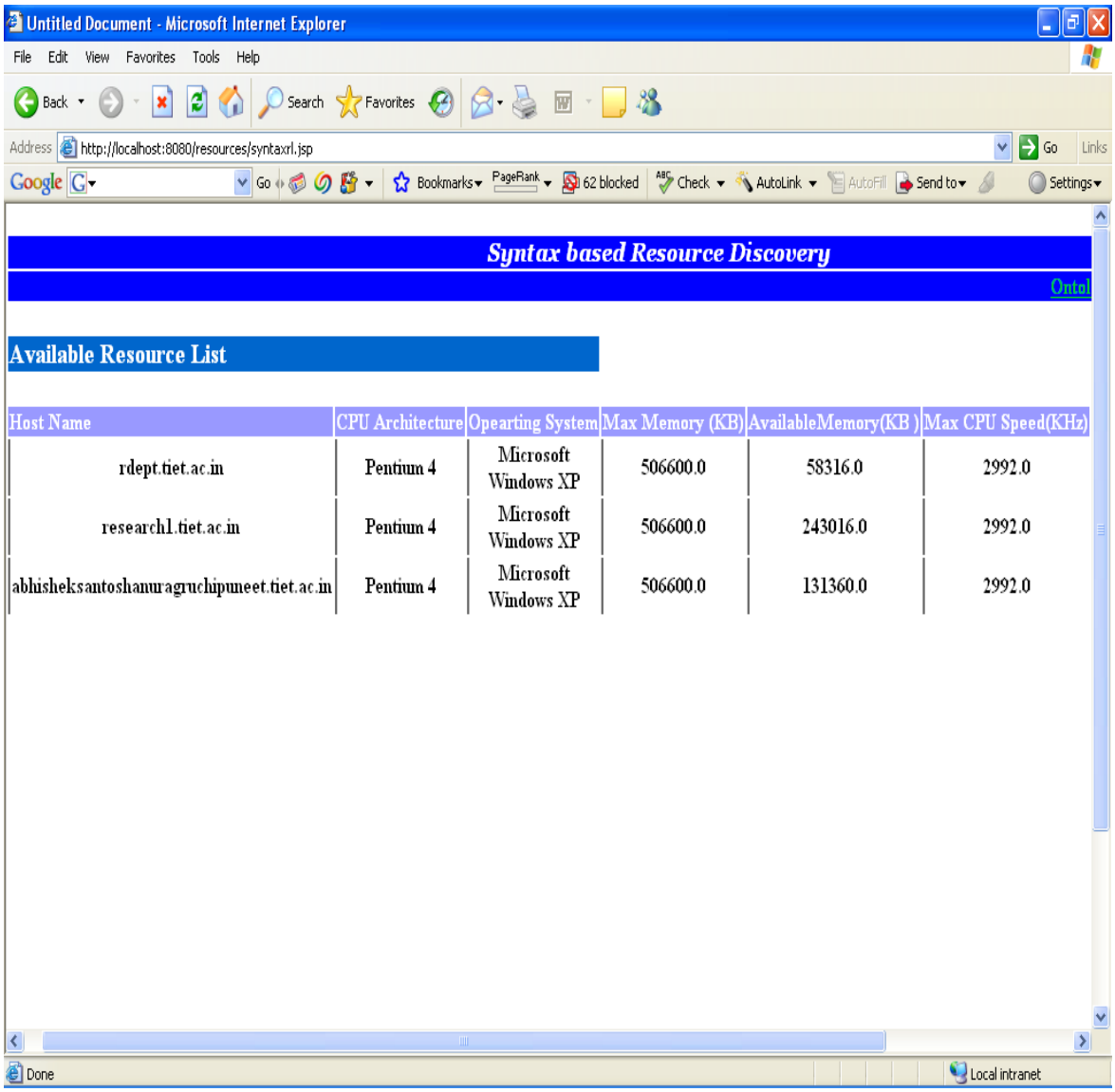
Query 1

CPU Architecture=X86

Available CPU Cycles (%) =80

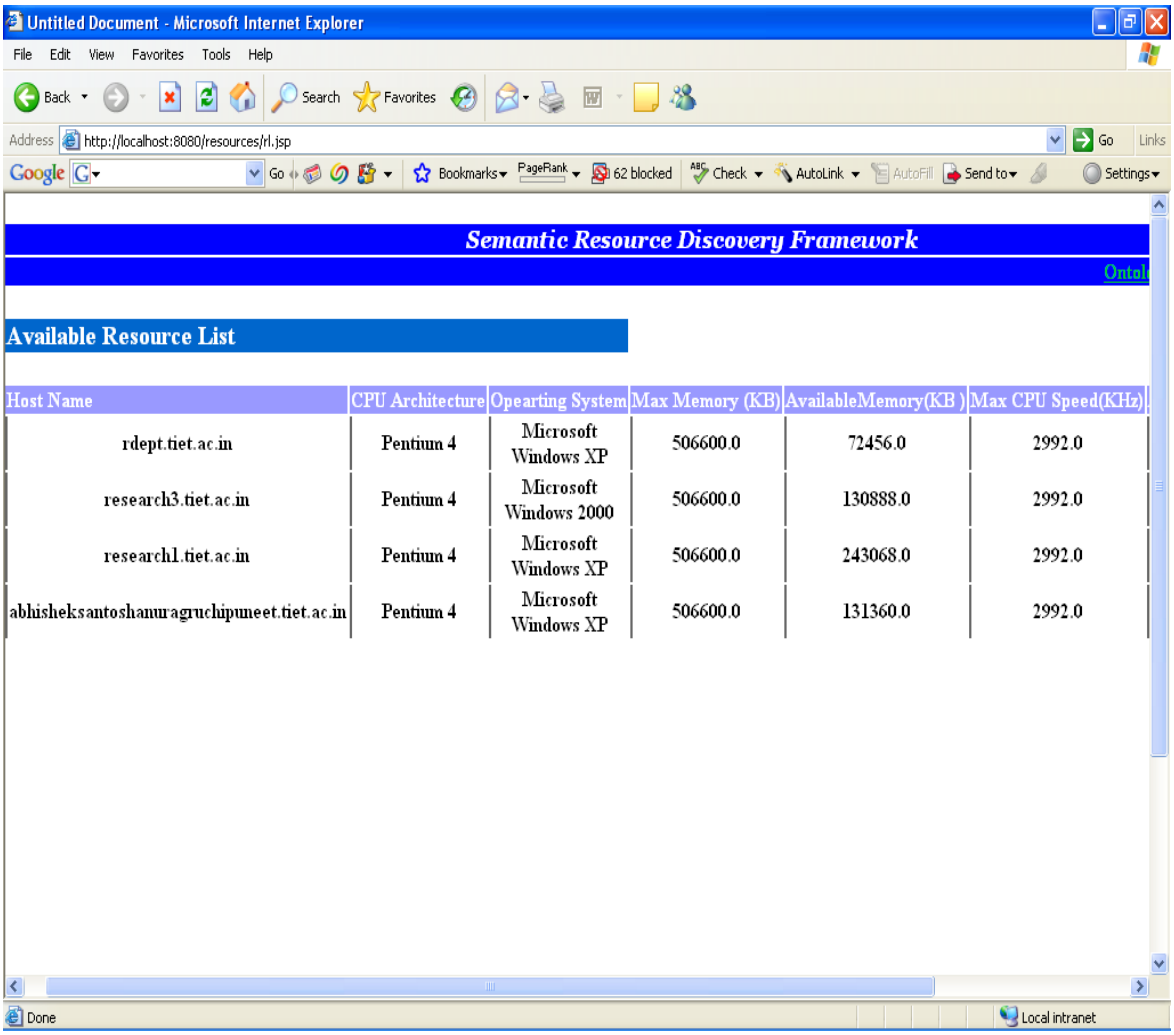
Memory=100 MB

Operating System=Microsoft Windows XP



Screenshot 6.2 List of available Resources: Syntax based Resource Discovery approach

Screenshot 6.2 shows the list of available Resources that fulfils user request i.e. machine with Pentium 4 CPU Architecture, 100MB available memory, 80% Available CPU Cycles of Maximum CPU Efficiency and Microsoft Windows XP Operating System. In this case only 3 machines fulfill user request.



Screenshot 6.3 List of available Resources: Ontology based Resource Discovery approach

Screenshot 6.3 shows the list of available Resources that fulfils user request i.e. machine with Pentium 4 CPU Architecture, 100MB available memory, 80% Available CPU Cycles of Maximum CPU Efficiency and Windows Operating System. In this case 4 machines fulfill user request.

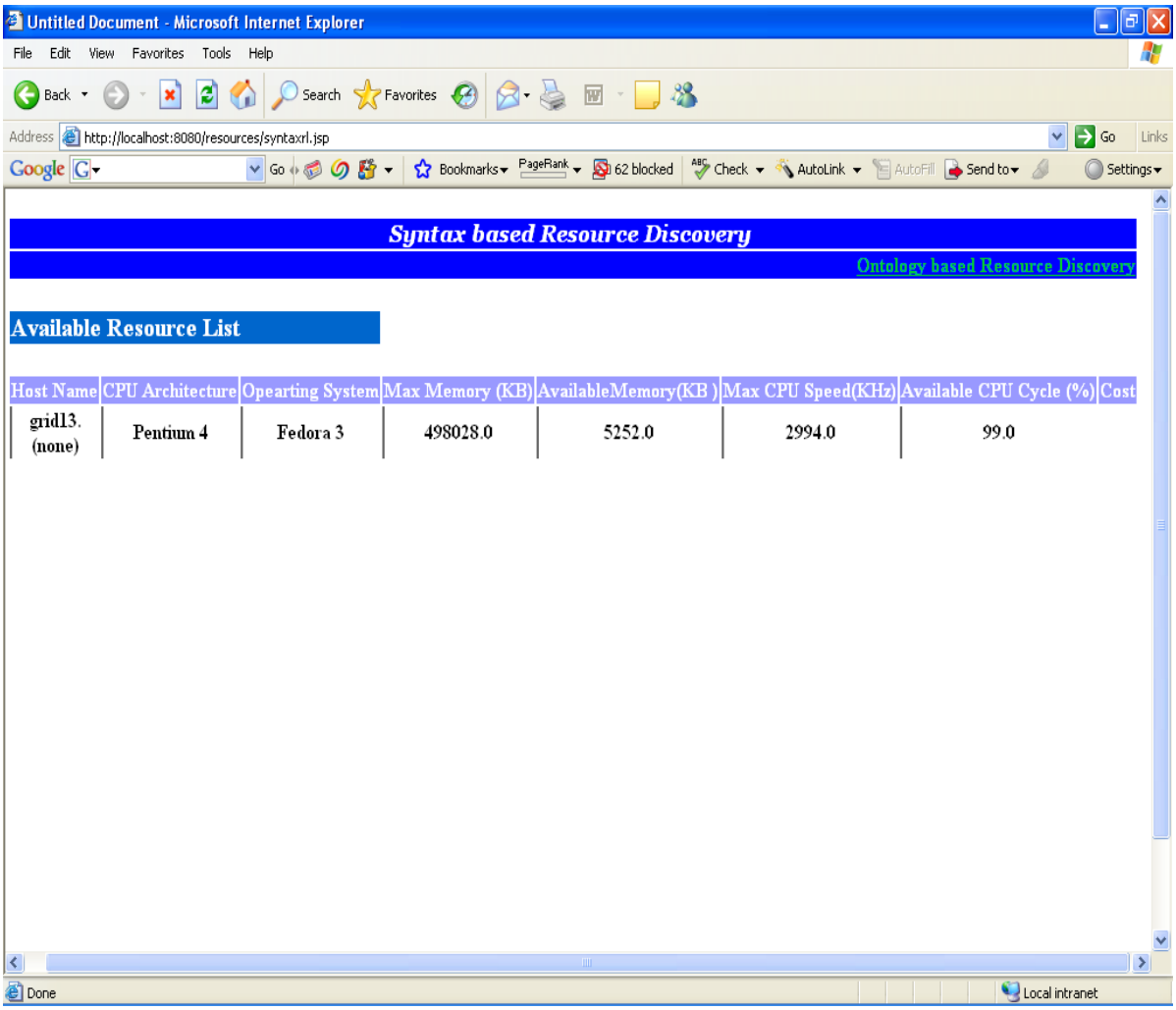
Query 2

CPU Architecture=Pentium 4

Available CPU Cycles (%) =70

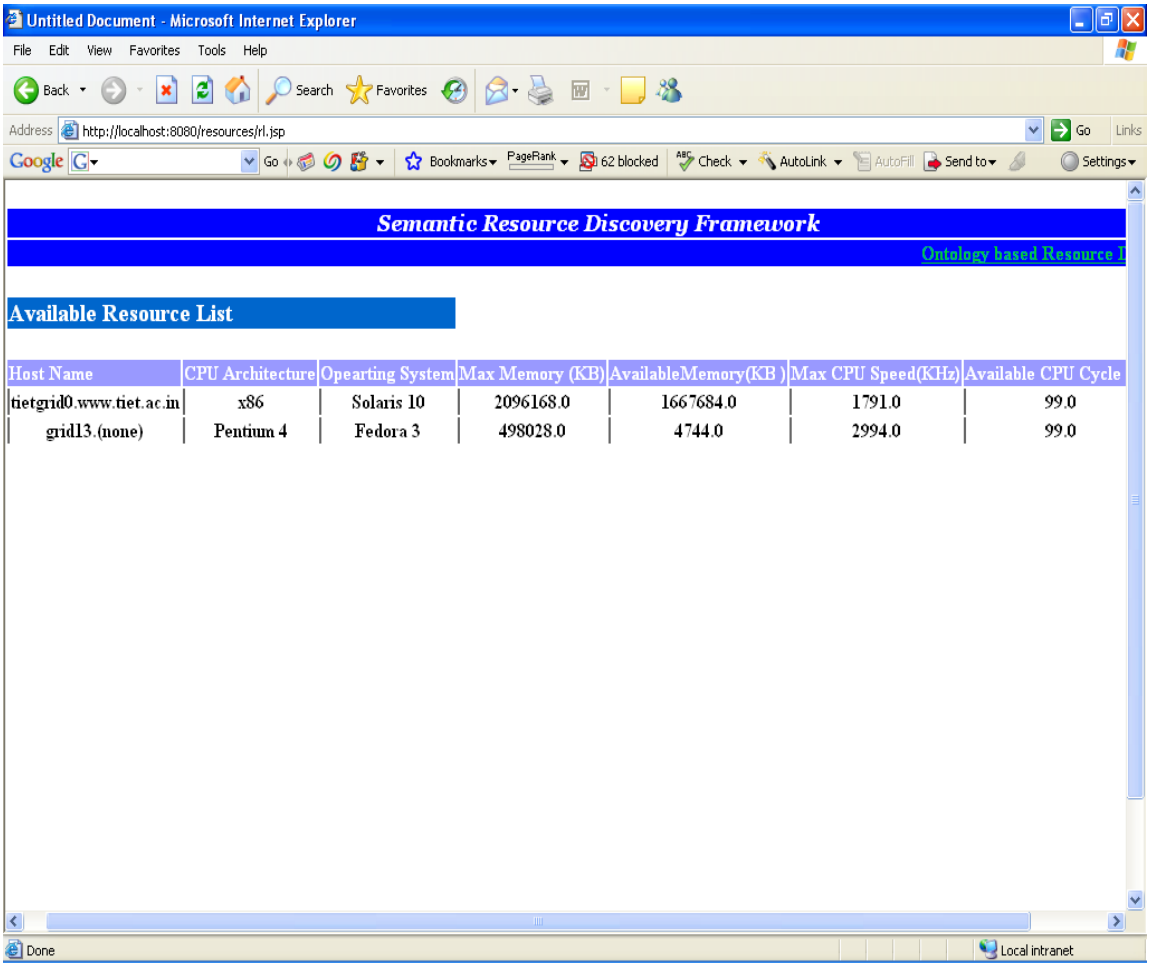
Memory=100 MB

Operating System=Fedora 3



Screenshot 6.4 List of available Resources: Syntax based Resource Discovery

Screenshot 6.4 shows the list of available Resources that fulfils user request i.e. machine with Intel CPU Architecture, 100MB available memory, 70% Available CPU Cycles of Maximum CPU Efficiency and Fedora 3 Operating System. In this case only 1 machine fulfill user request.



Screenshot 6.5 List of available Resources: Ontology based Resource Discovery approach

Screenshot 6.5 shows the list of available Resources that fulfils user request i.e. machine with Intel CPU Architecture, 100MB available memory, 70% Available CPU Cycles of Maximum CPU Efficiency and fedora 3Operating System. In this case 2 machines fulfill user request.

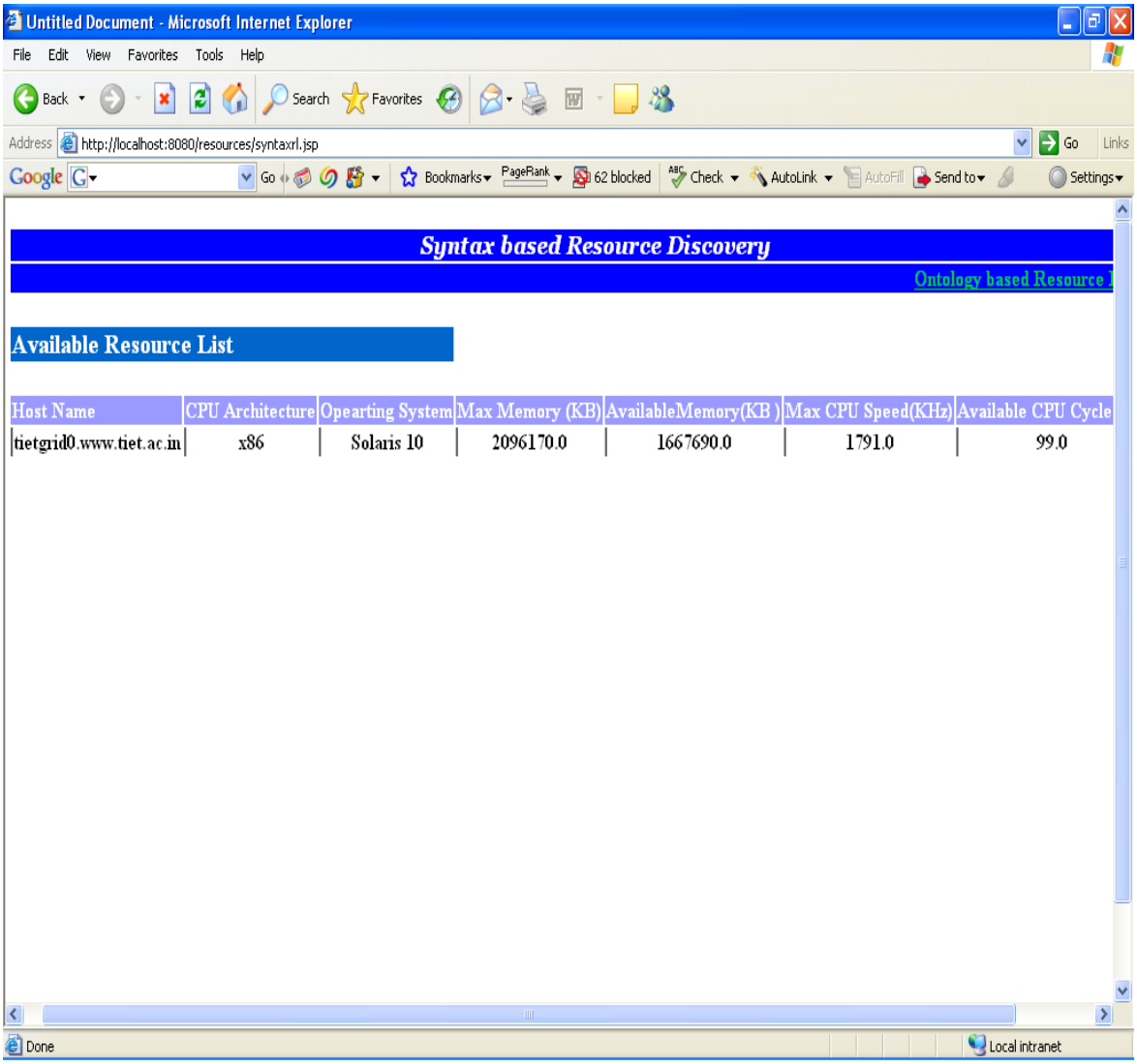
Query 3

CPU Architecture=x86

Available CPU Cycles (%) =70

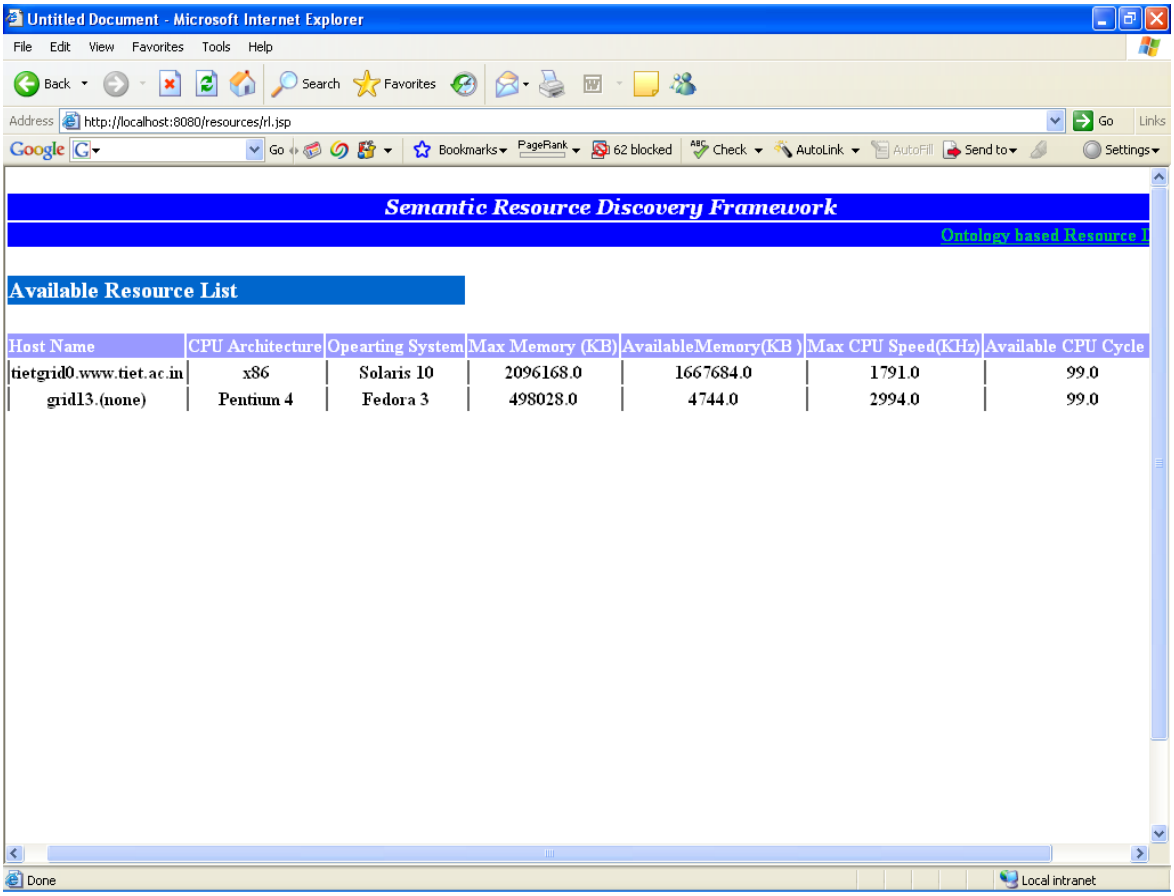
Memory=100 MB

Operating System=Solaris 10



Screenshot 6.6 List of available Resources: Syntax based Resource Discovery

Screenshot 6.6 shows the list of available Resources that fulfils user request i.e. machine with x86 CPU Architecture, 100MB available memory, 70% Available CPU Cycles of Maximum CPU Efficiency and Solaris 10 Operating System. In this case 1 machine fulfill user request.



Screenshot 6.7 List of available Resources: Ontology based Resource Discovery approach

Screenshot 6.7 shows the list of available Resources that fulfils user request i.e. machine with Intel CPU Architecture, 100MB available memory, 70% Available CPU Cycles of Maximum CPU Efficiency and Solaris 10 Operating System. In this case 2 machines fulfill user request.

This chapter focused on implementation of Ontology based Resource Discovery Approach that displays the resources having similar attribute value and not just exact keyword matches.

The next chapter summarizes this thesis work and suggests features that can be incorporated in future for an enhanced distributed decentralized Ontology management in Grid environment.

Chapter 7

Conclusions and Future Scope of 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 TU Grid Resource Ontology and implementation of an Ontology based Resource Discovery Approach as a solution to the problems in traditional approaches of Resource Discovery.

7.1 Conclusions

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. Correlation of this kind of information using a semantics based approach is the topic of this thesis.

Approaches to Resource Discovery are broadly classified as query based and agent based. 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.

Most of the State of the art Middleware such as Globus, Condor etc use query based approaches where Resource discovery is done based upon symmetric attribute-based keyword matching. These systems lack the ability of inexact matching. In the real world Grid user and Resource Provider wants to express the Resource description in their way. Thus, semantic interoperability is needed to go beyond simple keyword matching of attribute terms and comprehend the deeper meaning.

In this thesis, we have attempted to achieve semantic interoperability by developing an Ontology of Grid Resources. A TU Grid Ontology has been designed that describes Class of Resources and relationships between them. In particular, the Ontology based approach

is better than the syntax based Resource Discovery approaches used today in the following ways:

- An Ontology based Resource Discovery provides a flexible approach for Resource Discovery, as it is not dependent on the syntax used for resource description by the different Middleware.
- It provides a degree of freedom to the user whereby a user can input the query according to his knowledge without having any prior knowledge about the exact names of the available Resources that is advertised by various resource providers.
- It gives better results by finding Resources that have similar concepts not just same descriptions. Thus it returns all the relevant Resources and not just the Resources with matching keywords.

7.2 Future Scope of work

The Ontology based Resource discovery for Grid environments described in this thesis can be further improved by incorporating a dynamic creation and management of distributed Ontology which is to date is manual processes that will automatically create update the collection of new Resources and available concepts (resources).

References

[1] Foster, I., Kesselman, C. and Tuecke, S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizationss," International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.
http://www.globus.org/research/papers/anatomy.pdf

[2] C. Kesselman. I. Foster. Computational grids. In The Grid: Blueprint for a New Computing Infrastructure., chapter 2. Morgan-kaufman edition, 1999

[3] Ian Foster "What is the Grid? A Three Point Checklist" Argonne National Laboratory & University of Chicago

[4] Hongsuda Tangmunarunkit, Stefan Decker, Carl Kesselman "Ontology-based Resource Matching in the Grid —The Grid meets the Semantic Web" University of Southern California

[5] Brooke, J., Fellows, D., Garwood, K. and Goble C., "Semantic matching of Grid Resource Descriptions," In Proceedings of the European Across Grids Conference, 2004, http://www.Grid-interoperability.org/semres.pdf

[6] Lord, P., Alper P., Wroe, C. and Goble, C., "Feta: A lightweight architecture for user oriented semantic service discovery", In Proceedings of The Semantic Web: Research and Applications: Second European Semantic Web Conference (ESWC 2005), Heraklion, Crete.

[7] Li, L. and Horrocks, I, "A Software Framework for Matchmaking Based on Semantic Web Technology", In Proceedings of the 12th international conference on World Wide Web (WWW'03), Budapest, Hungary, May 2003.

[8] Raman, R., Livny, M. and Solomon, M., "Matchmaking: An extensible framework for distributed resource management", Cluster Computing: The Journal of Networks, Software Tools and Applications, 2:129-138, 1999

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

[10] Jean-Christophe Durand Grid Computing 'A Conceptual and Practical Study' November 8, 2004

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

[12] Ann Chervenak and others, “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets”, Journal of Network and Computer Applications, 2001.

[13] Brooke, J., Fellows, D., Garwood, K. and Goble C., “Semantic matching of Grid Resource Descriptions,” In Proceedings of the European Across Grids Conference, 2004, http://www.Grid-interoperability.org/semres.pdf

[14] David De Roure and others, “The Semantic Grid: Past, Present and Future”, Proceedings of the 2nd Annual European Semantic Web Conference (ESWC2005), Volume 93, Issue 3, 2005.

[15] Hai Zhuge “Special section: Semantic Grid and knowledge Grid” 17 July 2006; accepted 20 July 2006

[16] John Hagel, III and John Seely Brown, “Service Grids: The Missing Link in Web Services”, A Working Paper, http://www.johnhagel.com/paper_servicegrid.pdf, 2002. (accessed: 18 September 2005)

[17] 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.

[18] Ferreira, L., Bieberstein, N., Berstis, V., Armstrong, J., “Introduction to Grid Computing with Globus,” Redbook, IBM Corp http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf

[19] Foster, I., Kesselman C., Nick, J.M., and Tuecke, S., “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration,” Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002, http://www.globus.org/alliance/publications/papers/ogsa.pdf

[20] Duane Nickull “Service Oriented Architecture Whitepaper” Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA 95110-2704 USA

[21] Jarek Nabrzyski,Jennifer M. Schopf,Jan W ¸ Eglarz “GRID RESOURCE MANAGEMENT State of the Art and Future Trends” ,Kluwer Academic Publishers,Boston/Dordrecht/London

[22] Oscar Corcho , Pinar Alper, Ioannis Kotsiopoulos, Paolo Missier,Sean Bechhofer, Carole Goble “An overview of S-OGSA: A Reference Semantic Grid Architecture”School of Computer Science, The University of Manchester, Manchester, UK Received 7 November 2005; received in revised form 8 March 2006; accepted 20 March 2006

[23] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran “A Taxonomy and Survey of Grid Resource Management Systems”, School of Computer Science and Software Engineering,Monash University,Melbourne, Australia. http://www.buyya.com/papers/gridtaxonomy-report.pdf

[24] Adriana Iamnitchi and Ian Foster, “A Peer-to-Peer Approach to Resource Location in Grid Environments ,” University of Chicago.

[25] Manfred Hauswirth, Roman Schmidt ,”An overlay network for resource discovery in Grids,” http://dip.semanticweb.org/documents/An- overlay-network-for-resource-discovery- in-Grids.pdf

[26] Tierney, B. et al., "A Grid Monitoring Architecture," Global Grid Forum, Lemont, Illinois, U.S.A., January 2002. http://www.gridforum.org/documents/GFD.7.pdf

[27] http://www.ggf.org/ggf_Grid_understand.htm

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

[29] Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C., “Grid Information Services for Distributed Resource Sharing,” In 10th IEEE International Symposium on High Performance Distributed Computing, (2001), IEEE Press, 181-194, http://www.globus.org/alliance/publications/papers/MDS-HPDC.pdf

[30] Vishal Iyengar, Sameer Tilak, Michael J. Lewis and Nael B. Abu-Ghazaleh, “Non-Uniform Information Dissemination for Dynamic Grid Resource Discovery”, Department of Computer Science,State University of New York at Binghamton, NY , USA

[31] Rajesh Raman, Miron Livny, Marvin Solomon, “Matchmaking: Distributed Resource Management for High Throughput Computing,”University of Wisconsin , Madison, WI .

[32] Reynolds, J., and Weider, C., “Executive introduction to directory services using the X.500 protocol,” RFC 1308, FYI 13, 03/12 92.

[33] http://www.globus.org/mds/

[34] Bellwood, T., 2002. Bellwood, T. et all; UDDI Version 2.04 API Specification, July 2002,http://uddi.org/

[35] Iamnitchi, A., Foster, I., Nurmi, D. C., “A Peer-to-Peer Approach to Resource Discovery in Grid Environments,” Proc. of the 11th Symposium on High Performance Distributed Computing, Edinburgh, UK, 2002.

[36] 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.

[37] 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.

[38] Foster, I., Roy A., Sander, V., Winkler, L., “End-to-End Quality of Service for High-End Applications,” Technical Report, 1999

[39] Plale, B., Dinda, P., and Laszewski, G. Von., “Key Concepts and Services of a Grid Information Service”, In Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems (PDCS 2002), 2002

[40] Raman, R., Livny, M. and Solomon, M., “Matchmaking: An extensible framework for distributed resource management”, Cluster Computing: The Journal of Networks, Software Tools and Applications, 2:129-138, 1999

[41] Harth, A., Decker, S., He, Y., Tangmunarunkit, H., Kesselman C., “A semantic matchmaker service on the Grid,” World Wide Web Conference 2004: 326-327.

[42] 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

[43] M. J. Litzkow and M. Livny. “Experience with the Condor Distributed Batch System”.IEEE Workshop on Experimental Distributed Systems, 1990.

[44] M. J. Litzkow, M. Livny, and M. W. Mutka. “Condor: A Hunter of Idle Workstations”. In Proc. of the 8th Int'l Conf. on Distributed Computing Systems, 1988

[45] Simone A. Ludwig and S.M.S. Reyhani, ”Semantic Approach to Service Discovery in a Grid Environment,” Cardiff University, Brunel University, UKSmith, W. and Gunter D., “Simple LDAP Schemas for Grid Monitoring”, Global Grid Forum, GWD-Perf-13-1, June 2001.

[46] Phillip Lord, Chris Wroe, Robert Stevens, Carole Goble, Simon Miles, Luc Moreau,Keith Decker, Terry Payne and Juri Papay ,“Semantic and Personalised Service Discovery ,” University of Manchester, University of Southampton,H UK.

[47] Brooke, J., Fellows, D., Garwood, K. and Goble C., “Semantic matching of Grid Resource Descriptions,” In Proceedings of the European Across Grids Conference, 2004, http://www.Grid-interoperability.org/semres.pdf

[48] Chen, L, Shadbolt, N, Goble, C, Tao, F, Puleston, C, and Cox, S.J., “Semantics-Assisted Problem Solving on the Semantic Grid,” Computational Intelligence, Vol.21, No.2, 2005, pp.157-176, http://www.geodise.org/files/Papers/ComputationalIntelligencePaper.pdf

[49] David De Roure and others, “The Semantic Grid: Past, Present and Future”, Proceedings of the 2nd Annual European Semantic Web Conference (ESWC2005), Volume 93, Issue 3, 2005.

[50] http://www.cs.wisc.edu/condor/manual/v8.2

[51] http://www.w3.org/2004/OWL/ -

[52] http://www.w3.org/RDF/ -

[53] http://www.MySQL.com

[54] http://www.protégé.stanford.com/

[55] http://www.sourceforge.net/jena

[56] http://www.tomcat.apache.org

[57] http://www.sourceforge.net/sigar

[58] http://www.java.sun.com/products/jsp/

List of Publications

1. Santosh Jaiswal, Inderveer Chana, Seema Bawa, “Hybrid Approach of Resource Discovery in Grid” accepted in International Conference on Emerging Trends in High Performance Computing Architecture Algorithms and Computing, HiPAAC 2007, SSN Collage of Engineering, and Chennai 11 July, 2007**(Accepted)**.

2. Santosh Jaiswal, Inderveer Chana, Seema Bawa, “Ontology based Resource Discovery in Grid”, published in National Conference on Information Technology: Emerging Engineering Perspective and Practices, ITEEPP’07 2007, Thapar University 6-7 April 2007 .