62216175

QA
76.9
.C58
S23
2010

WEB SERVICE-BASED GRID RESOURCE DISCOVERY

by

Saadat Bokhari, Syed
M. Sc. Computer Science University of Karachi, 1989

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2010

Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in parts, at the request of other institutions or individuals for the purpose of scholarly research.

WEB SERVICE-BASED GRID RESOURCE DISCOVERY

M. Sc. Computer Science, 2010

Saadat Bokhari, Syed

Department of Computer Science

Ryerson University

ABSTRACT

This thesis proposes a simple and scalable web-based model for grid resource discovery for the Internet. The proposed resource discovery model contains the metadata and resource finder web services. The information of resource finder web services is kept in the repositories that are distributed in the application layer of Internet. The resource finder web services will be discovered by sending queries to the repositories in a similar way as the DNS protocol. The underlying technology for implementation of the two architectures of this model is introduced.

These architectures are Direct and Centralized Web-Based Grid Resource Discovery Applications. The resource discovery time of adding each of these two models on the top of GridSim is computed.

By doing the scalability test, we found that when increasing the load of grid with more users and resources the cost of our model in comparison to the grid resource discovery time is marginal.

ACKNOWLEDGEMENTS

I have a deep feeling of being indebted to my dedicated, consistent, innovative and hardworking supervisor, Dr. Abdolreza Abhari, for his guidance with patience and encouragement on my research. It was indeed extremely hard to accomplish my task without his support and positive criticism.

I feel grateful to my co-supervisor Dr. Alexander Ferworn for his constant supervision, prompt decisions and supports that led me to success.

I am also grateful to Dr. Alireza Sadeghian for his timely help and support during my study and research. I am thankful to Dr. Marcos Escobar for his help and guidance in the field of statistical analysis and Ms. Carol Sevitt for her help in English editing.

I am thankful to all advisory committee members for their criticism and guidance.

I also want to acknowledge my colleagues in the Distributed Systems and Multimedia Processing Laboratory especially Mr. Yijun Chen for all his help.

I would also like to thank my wife and all of my children who helped me in their own way to achieve my goal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ACRONYMS

ACRONYMS	DEFINITIONS
ASL	Average Search Length
BOINC	Berkeley Open Infrastructure for Network Computing
CE	Compute Element (Processing Computer)
CM	Control Manager
CWGRD	Centralized Web-Based Grid Resource Discovery model
DNS Protocol	Domain Name Server Protocol
DWGRD	Direct Web-Based Grid Resource Discovery model
ETC	Expected Time to Compute
GIS	Grid Information Service
GIS Node	Node Managing Resources
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IE	Instrument Element
jUDDI	Java based Universal Description, Discovery, and Integration Registry
Metadata	Administrative data used for indexing
OGSA	Open Grid Services Architecture
P2P model	Peer to Peer Computers Network model without using central server
RCT	Resource Category Tree
Resource Finder	A web service focusing on resource discovery in our model
SE	Storage Elements

SOAP	Simple Object Access Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator (web address)
VCM	Virtual Consistency Manager
VCR	Virtual Control Room
VO	Virtual Organization
VEGA	Versatile services with Enabling intelligence based on knowledge, Global uniformity and Autonomous control
VIG	VEGA Information Grid
WSRF	Web Service Resource Framework
WGridSP	Web-based Grid Scheduling Platform
XML	Extensible Markup Language

CHAPTER 1

1. MOTIVATIONS AND OBJECTIVES

In this chapter, we explain the motivations and objectives of our research in terms of rationale and problem statement, hypothesis, scope/goal and importance of the topic of this thesis. The rationale and problem statement are given in subsection 1.1. The hypothesis and assumptions are described in subsection 1.2. The scope and goals of research are mentioned in subsection 1.3. The importance of the research is elaborated upon in subsection 1.4.

1.1. RATIONALE AND PROBLEM STATEMENT

With the popularity of the Internet, there is a strong need for grid computing applications to share their resources through the Internet. For example as described by Szalay *et al.* [1], the World-Wide Telescope application, which is a project for sharing astronomy data, can accomplish its objectives by using grid computing infrastructure through the Internet. Therefore, the main motivation of this work is providing an effective method to Internet users for accessing the grid resources.

For the Internet users who want to access the grid resources, the required resources should be found through the proper way of contacting grid infrastructures. Considering the spread of Internet access, the number of users who want grid resources is growing rapidly. Therefore, it is important for the grid resource discovery model to have the scalability characteristic for finding and accessing the resources. There have been many solutions for providing an efficient resource discovery model that will be discussed in detail in Chapter 2. However, most of these grid resource discovery models are not focused on the scalability issue when the goal is accessing the grid resources through the Internet.

Coulouris *et al.* define scalability of a system as "A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users" [2]. According to Nueman [3], the scalability of the system can be discussed in terms of the three attributes of numeric size (i.e. number of users and resources), geographical size and administrative capability of that system. Nueman [3] suggests that using caching, replication and distributing the information of the resources are techniques for providing scalability for the growing system in any of the three dimensions of size, geographical and administrative information requirements.

Considering the importance of scalability for accessing grid resources through the Internet, we have to address two separate problems in this thesis:

1- Providing the scalable solution: A scalable solution should be provided with respect to all three issues mentioned by Nueman [3]. The scalable solution is required for providing Internet access to grid resources. This solution should remain effective when the number of users and resources increase along with geographical expansion of the system. This solution should also be capable of addressing the administrative information requirements when the system is growing. Specifically, this scalable information administration means providing effective methods for searching and storage of the information of the grid with increasing resources in Internet.
2- Examining the scalability of a proposed solution: Another problem in our work is that we need to test the scalability of our model by increasing the number of users and resources and examining the effectiveness of information administration of the proposed model. So we need to measure the effectiveness with proper performance metrics.

To address the first problem, we believe that using web services can solve the size and geographical expansion problems and can provide a scalable and reliable interface for Internet users to access grid resources. Web services are not dependent on any computing platform or programming language and are using SOAP [4] messages with XML format [5] for communication. SOAP messages uses HTTP protocol and provide a reliable and secure way of communication in the application layer of Internet as described by Coulouris *et al.,* in [2]. Our solution is proposing the architectural model that employs Web services as the user interface to provide size and geographical scalability for the Internet access to grid resources.

We are presenting distributed information storage together with distributed search solution for grid resource discovery to address the administrative information requirements of the scalability problem.

To address the second problem, we found that testing the scalability for the real grid on the Internet is a hard and expensive task. Thus we need to simulate the grid computing to be able to examine scalability by increasing the number of users and resources. Grid simulation is also a difficult task because a grid is not similar to the usual networks. The network simulation tools such as ns2 [6], which is currently the popular network simulator software that is developed for the test of networks, is not useful for grid simulation; therefore, we need to find suitable grid simulation tools.

For testing the scalability of our model, we need to examine the effectiveness of the information administrative approach that is used by the model. Total grid resource discovery time is the performance metric that we used to measure this effectiveness. We used a DNS-like solution for administrative information requirements that is a scalable model under the increasing load and size of the Internet. The reason of scalability of DNS protocol as described by Mockapetris [7] is employing distributed storage of information across many geographical points. This approach avoids a single point access of information that can become a bottleneck as explained by Tanenbuam *et al.* in [8]. The distributed search resulting from a DNS-like solution will be explained in the hypothesis section in more detail in section 1.2 of this chapter. Thus we narrowed down the examination of the scalability problem in our simulation only to find the additional time of information administration time of our model (i.e., cost) on grid resource discovery time under increasing load of the grid.

Also there is another part for addressing the second problem. For measuring the cost of the model, we have to simulate web service which is the Internet user interface of our solution. We did research on several network simulation tools, and in particular, we studied ns3 [9] which is the latest version of network simulation tools after release of ns2. We found that it is not possible to simulate web services with ns3. To the best of our knowledge, there is no simulation tool that can be used for simulation of web services together with grid computing. So, one of the challenges of this work was finding a grid simulation tool that can be used together with the implementation of the web services in the simulation study of the whole system to test the scalability of the proposed model.

1.2. HYPOTHESIS

Our goal is providing a scalable access for the user being able to access the grid resources in the application level of the Internet by using web services interface located on the top of TCP/IP. The distributed information storage and search employed by the model is very similar to Domain Name Servers (DNS) protocols described by Mockapetris in [7] and [10]. DNS Protocol is explained below in more details.

Krishnamurthy *et al.* [11] has referred to DNS protocol as a one of the application level protocols underlying HTTP protocol that maps URL to IP addresses. In DNS protocol name servers keep the zone files that have the distributed information of URLs and IP addresses based on the domain of the URL of each of the Internet resources. A hierarchical naming schema is used in DNS for distributed storage of the information of Internet URLs and IP addresses. In this tree-like structure, each name server has a zone file containing the information of URLs based on a specific domain that can be geographical, educational or some other category. Name servers use the queries to perform the distributed search and also perform caching to speed up the process of mapping URL and IP addresses. Considering the performance of the name servers, if resolving the queries is performed locally by cache, it is in the hundreds of milliseconds, and if the requested URL is not in the cache, the queries will be resolved in a couple of seconds.

We assume the proposed model is similar to DNS protocol and uses the same techniques such as caching and distributed search for providing scalability. Our model uses the queries for searching the URL of the web services similar to DNS search for URLs. Instead of name servers our model uses repositories with similar zone files for keeping the information of the URLs of the resource finder web services and their related grid resources.

To clarify how our model works, consider the following example and assume our model is implemented in the network of Ryerson University that we referred to as a "region" for a user who wants to access the grid resources in this network. Suppose this user is in Ryerson University and knows the URL of a web service of the Lab A. This service allocates user's required grid resource which is the server B in Lab A.

Assuming the URL of the Lab A is "www.LabA. ryerson.ca" Calling this URL the web service of Lab A is triggered and the user can access the required resource by negotiation with this service.

In the second scenario, the user does not know the URL of the web service of Lab A but knows the URL of central repository (UDDI) of Ryerson University. Assume the URL of this repository is "www.RyersonUDDI.ryerson.ca" and that it has the information of the services and related resources of all the labs in Ryerson University region. In this case the user contacts the repository and gets the URL for calling the web service of Lab A. In the third scenario, the repository of Ryerson University region does not have any service that provides the required grid resource. Therefore, this repository sends DNS-like queries and uses distributed search by communicating with other repositories to find the required resource from other universities or other regions.

We assume the grid resource finder service is able to access the grid infrastructure. We also assume it has all the standards for Open Grid Service Architecture (OGSA) described by Foster *et al.* in [12] for performing the grid tasks. It can find the resources from its list referred to as metadata, which is assumed to have the latest information on grid resources. We also assumed after finding the URL of the grid resource finder, then that service will act as a resource broker and allocate grid resources upon request. Also we assume a UDDI technology, such as the one presented by Benson *et al.* [13], exists and that it can be used for grid service discovery.

To deploy distributed search efficiently, we assume the UDDI repositories have a caching mechanism to speed up the distributed search process. We assume in the worst case scenarios similar to a DNS search, that if the requested URL is not available in the cache, it can be found in a matter of seconds. To test whether our model is scalable or not, we should show that similar to a DNS search, the information administration time, which is the searching time that is added by our model (because of using web services), is negligible compared to the grid resource discovery time when the numbers of users and resources of the grid are increasing in the Internet.

1.3. SCOPE AND GOAL

The main goal of our research is to provide a scalable solution for Internet users to discover and access resources in the grid environment. At the end of this section, we will explain if we reached to this goal or not, but first we will discuss the scope of the work.

Although our model uses the same mechanism as DNS protocol in the Internet application level, we do not focus on the lower layers of the grid in the proposed model. For example, we do not provide any ways to update the information of grid resources when they are changing. We assume the resources information is updated periodically by grid providers or regional resource managers.

We do not implement the architectural model in the real physical grid environment, but we use it together with the simulated grid environment. We will only implement the best case scenarios of our model for regional searches because of the complexity of DNS-based distributed search that we proposed in the model. Our architectural model is using caching technique in its repositories similar to DNS protocol, and that is why searching time in our model after the first time will be equal to the best case scenarios in the second search for the resource finder web service. Our model will first search for resources from the regional repository, and if the required resource is not there, then another repository outside of the regional domain will be contacted by using a distributed search approach.

According to the assumptions, we do not discuss the structure of resource finder web services and we do not discuss the internal structure of UDDI in this work. We also assume a tree structure of repositories with a hierarchy similar to the domain name services in DNS protocol is also available for our model. We will introduce the underlying technology to implement our model for the resource discovery scenarios, but we will not go into the details of the whole hierarchical model of repositories.

Finally simulation of the grid together with the prototype of the proposed architecture on top of that is the methodology to estimate the cost of our model in terms of addition to grid resource discovery time while increasing the load of grid resources and users to test the scalability. The statistical analysis with the required assumptions will also be used to analyze the simulation result to show the differences between the average resource discovery times of our model in comparison to the average resource discovery times of the grid itself.

We found that when increasing the load of the grid, the cost of our model decreases, and finally with the grid load of 100 users and 100 resources and more, the cost of our model in comparison to grid resource discovery time is marginal. Based on this result, we conclude that if our model were implemented on the Internet with thousands of grid resources and users, it would be scalable. Even if we add the couple of seconds for DNS-like distributed search to include the worst-case scenarios to the cost of our model, still this time that is the time for administration of information when the system is expanding, will be negligible compared to resource discovery time of grid in the Internet. These results support the scalability of our model based on the assumptions that we have made.

1.4. IMPORTANCE

The popularity of the Internet has motivated the grid providers to move their current grid projects such as Word-Wide telescope [1] under the TCP/IP network. Similar to the electricity grid, in the near future these grid resources will be available for everybody. The standards for building the new concept of grid services released for OGSA that is presented by Foster *et al.* in [12] and promising technologies such as UDDI version 3.0 that is capable of discovering grid services as mentioned by Benson *et al.* in [13] show the future direction of grid computing is based on the Internet.

For Internet access to the grid resources having scalable resource discovery methods is required. There are limited works that discuss the scalability of the methods for Internet-based grid computing. Regarding scalability in this work, we presented an approach for effective administration of the information of the grid resources that is capable of handling increasing numbers of Internet users and grid resources. So we have summarized the problems and our solutions for providing the scalable method for grid access by Internet.

Now the question is: why do we need scalability for Internet-based grid computing? As discussed before, for providing Internet access to grid resources which is the motivation of this work, we need to provide scalable distributed resource discovery. Otherwise, if the solution will be based on central approach for resource discovery (for example by having a central database registry of all resources), whenever the number of users and resources in the grid increases, this central database registry become a bottleneck in the system as mentioned in [8]. Therefore, using non-scalable solutions is not possible for grid computing when we want the grid resources to be accessible through the Internet.

The rest of our thesis is structured as outlined below:

Chapter 2 introduces recent and past research works done on the grid and especially on grid resource discovery. Chapter 3 discusses the web-based architectural model and implementation strategy for a direct and centralized web-based architecture. Chapter 4 elaborates on the simulation package GridSim, including the simulation results of direct and centralized web-based architecture. The discussion of scalability and statistical analysis are also given in Chapter 4. The conclusions are drawn, our contributions are reiterated and potential future work is discussed in Chapter 5.

CHAPTER 2

2. RELATED WORKS

This chapter discusses works related to general grid computing and grid resource discovery. It mentions one sample for each work on general grid computing but focuses more on resource discovery-related works. Chapter 2 is further divided into section 2.1, section 2.2, section 2.3 and section 2.4. The general grid computing-related works are mentioned in section 2.1. Section 2.1 comprises five different related works in the area of general grid computing. Section 2.2 also contains different resource discovery-related works. The layered architecture of resource discovery is presented in subsection 2.2.1. Subsections 2.2.2 and 2.2.3 discussed hierarchical and tree based model discoveries. Subsection 2.2.4 presents Grid Resource Discovery based on Semantic Information that describes P2P and Galaxy models. Finally UDDI-based Resource Discovery is given in subsection 2.2.5. The numbers of grid simulation packages are mentioned in section 2.3. The comparison of our work with other architectures designed for resource discovery is given in section 2.4.

2.1. GRID COMPUTING-RELATED WORKS

2.1.1. Scheduling in Grid Systems

As mentioned by Xhafa *et al.* [14], the inefficient scheduler system in middleware makes it difficult to bring real-world applications such as search, education, e-commerce, collaborative work, file storage and high performance computing for efficient utilization in a grid environment. The five classical batch job scheduling is evaluated by Xhafa *et al.* in [14] for grid computing. The scheduling system maintains a vector (n, m) where n stand for number of jobs and m means number of machines. The Expected Time to Compute (ETC) for each job is calculated by a batch job scheduling system. The batch methods are evaluated by computing the concluding time of the latest job and the sum of the concluding times of the jobs already performed. The batch job methods are also evaluated by measuring the quality of schedule with respect to the utilization of resources and allocation of the jobs to the best resources in terms of speed.

The benchmark simulation model is used for testing the implementation of batch methods and described in [14]. The programs for batch job methods are written in C++. The 512 jobs on 12 computers are used to simulate a grid environment. The information of resources is given in advance to the simulation model. The computational results reported in [14] show that the presented batch methods do not perform well. The performance of batch job methods depends on high or low diversity in the nature of jobs and the grid scenarios. The performance also depends upon the volume of different kinds of resources. The performance further depends upon consistency level of the computing resources that may be consistent, semi-consistent or inconsistent. The final conclusion in [14] is for designing an efficient batch job scheduler for a grid, the grid characteristics should be known in advanced.

2.1.2. Replica Management in Data Grids

Belalem *et al.* [15] extended the idea of providing efficient access to the data without conflicts in a large-scaled grid by use of replication technique. Replication is a technique for producing the exact copy of the original file. The consistency of replica is an important issue in data replication. The consistency of replica is the degree of similarity with the master file after updating the replica from a master file. There are two approaches to manage the consistency of replicas. The first approach is pessimistic replication that updates all replicas synchronously. The second approach is optimistic replication that updates single replica at a time. Belalem *et al.* [15] have proposed a third approach named hybrid method by joining the pessimistic and optimistic approaches. The hybrid approach consisting of a tree-based hierarchical model, manages a large-scaled grid into two levels for the consistency management of replicas. The whole grid comprises sites defined at level 0. Each site has a computer or a network of computers. Any site selected for replication is defined at level 1. The level 1 is divided into Intrasite and Intersites tasks. The management of replica consistency using optimistic approach by Virtual Consistency Manager (VCM) within a site at level 1 is called Intrasite. The VCM introduced in the paper [15] is software to manage replica consistency at each site. Each VCM for a site cooperates with other VCMs of other sites in the whole grid. The management of replica consistency using a pessimistic approach among different sites in the grid through their respective VCMs is named Intersites.

The OptorSim presented by Bell *et al.* [16] is used for simulation of a hybrid approach and its components. The simulation parameters used by Belalem *et al.* [15] are number of sites, replicas and requests. The results indicate that the quality of service is better in a hybrid than in an optimistic approach. Also the response time is better in a hybrid than in a pessimistic approach.

2.1.3. Integration and Sharing of Resources

For integration and sharing of resources, Li *et al.* [17] introduce an information-based grid consisting of a database called VIG. VIG stands for VEGA Information Grid while VEGA stands for Versatile services with Enabling intelligence based on knowledge, Global uniformity and Autonomous control. VIG structure consists of virtualization-based relation schema, virtual database and loosely coupled interface architecture. These are explained as the follows:

- Relation Schema: Contains information of physical resources such as computers in the networks (defined by resource providers), a data model for business requirements and a personalized environment for user requirements (defined by user).

- Virtual Database Engine: Is designed to investigate the result of user queries sent to multiple resources.

- Interface Architecture: Takes care of interaction between relation scheme and virtual database engine.

To test the performance of the database four queries were tested that resulted in an average user time of 1.8 minutes.

2.1.4. Desktop Grid Applications

Desktop computers are used all over the world. Schmidt [18] recommended the formation of a grid consisting of desktop computers to provide immense computing power for scientific applications at very low cost. The volunteer desktop computers in the grid can be effectively used by applications with no or very little cost. Schmidt [18] discusses the example of Desktop Grid is based on the middleware platform known as Berkeley Open Infrastructure for Network Computing (BOINC). The BOINC middleware is the open source software consisting of servers, clients, databases and modules. The server middleware consists of data and scheduler software. The server uses a HTTP-based protocol. The different units of software communicate with each other to distribute, process, and return jobs. The author argues that Desktop Grid Application will benefit research and user groups.

The computing power of a desktop grid described by Schmidt [18] is enhanced by using GPU-based desktop computers instead of CPU-based desktop computers. GPU stands for Graphics Processing Unit. The GPU has been developed for computer games, which are lively in nature and extremely competitive. It is shown by integrating GPU into BOINC project that the speed of GPU-based dedicated desktop grid computers becomes significantly faster than that of CPU-based dedicated desktop computers.

2.1.5. Instruments Control in Grid

A grid infrastructure as mentioned by Lelli *et al.* [19] may consist of the data acquiring instruments like sensors that can be heterogeneous devices and geologically distributed. These instruments should be remotely operated and monitored by the teams at different locations using a grid. The goal of the work presented by Lelli *et al.* [19] is providing the software tools for interaction of data acquisition instruments and other parts of the grid. The classification of data acquiring instruments is taken into account in the design of new software that can be used as the grid component to control these instruments. A general model called uniform model consisting of generic instruments is presented by Lelli *et al.* [19]. The software component called Instrument Element (IE) consisting of different services is also introduced in [19]. It enhances the existing grid to control real scientific instruments remotely. The services in Instrument Element provide functionalities for monitoring, configuring, controlling and accessing of instruments remotely.

The first release of IEs is implemented in Java language and Java-based components. The scalability and flexibility of the first release of Instrument Element is tested through the "command reception" and "distribution performances" experiments. The command reception is testing the receiving capability of the Control Manager (a component of IE). The command distribution is the capability of distribution of command inside IE internal components. Two tests have been performed to check the receiving capability of IE middleware. The Control Manger receives a request from a client. The Control Manager replies 50 responses per second on average to the requests received from clients. The second test is concerned with command distribution by measuring message handling. These results of the tests show scalability and flexibility of IE. The first release of IEs software is currently employed in several grid projects that are mentioned by Lelli *et al.* in [19].

2.2. RESOURCE DISCOVERY-RELATED WORKS

The sharing and integration of heterogeneous resources for computational and data storage projects has become an important area of grid research. The resources on the grid systems are more widely distributed and heterogeneous than that on traditional and cluster systems. The resource discovery is a key grid management tool for extraction of resource information in the grid environment. The resource discovery management tool in the grid is based on resources organization. Most approaches of resource discovery for grid computing treat resources equally. These approaches are facing problems in terms of efficiency in response time and the need to generate complex queries. Therefore resource discovery architecture is needed that should be simple and scalable to discover resources. Some of the works already done on the subject of resource discovery models are discussed next.

2.2.1. Layered Architecture-Based Resource Discovery

The layered architecture of grid computing was proposed by Foster *et al.* [20] in 2001. It includes five layers as shown in Figure 2.1: Fabric, Connectivity, Resource, Collective and Application layers. The Fabric layer is the first layer, handling local resources at a particular site. The Connectivity layer uses network communication protocols for data transfer between resources. The Resource layer uses protocols of the connectivity layer and interfaces provided by the fabric layer to manage a single resource. The collective layer holds the information of multiple resources and manages resource discovery, task scheduling and the allocation of service. The collective layer discovers resources based on the information from lower layers. Finally, the application layer is the closet layer to the user and provides applications within the virtual organization to access the grid computing.

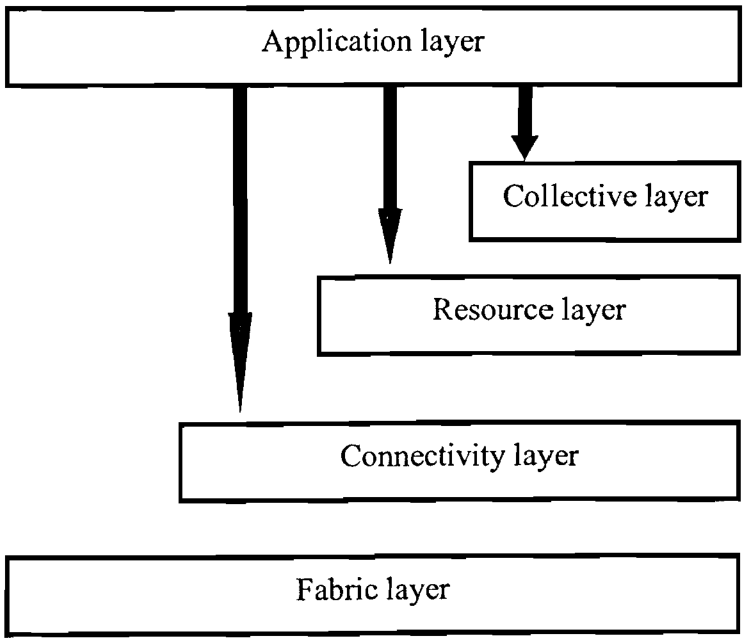


Figure 2.1 Layered Architecture from [20]

2.2.2. The Grid Resource Discovery Method Based On Hierarchical Model

Yin *et al.* [21] has proposed a new model called the Hierarchical Resource Organizational Model. The model consists of the three layers to keep the information of a grid. These three layers are Physical Network, Resource Information and Index Information. The Physical Network Layer is lowest level containing the physical resources linked with each other on the Internet. For each resource there is an information node called resource node that is placed in the middle layer named Resource Information layer. Therefore the middle layer contains virtual organizations (VO) information which is the group of resource nodes with star topology that has a super node in the center. The Super Node keeps all the information regarding the resources of VO as the adjacent lists. These Super Nodes are used in the Index Information Layer which is the highest level for resource discovery. This layer contains information of all Super Nodes of the middle layer which is constructed in a ring topology structure and will be used for hierarchical resource discovery as presented in [21].

Three simulation tests performed by Yin *et al.* [21] are using 100, 500 and 1000 resource nodes on the three kinds of discovery models. They compared their model against two other models that they called Exhaustive and Lumped models. The exhaustive model searches all the resource information nodes and Lumped model uses only a single point for resource discovery. As explained in [21], Hierarchical model not only outperforms the two other models regarding the resource discovery time but also does not have the performance problems of the two other models. The single point based resource discovery used in the Lumped Model can be a bottleneck and source of failure. Also blindly search discovery model used in Exhaustive Model does not scale when the resources are increasing.

2.2.3. Tree Structure-Based Resource Discovery

Sun *et al.* [22] proposed Resource Category Tree (RCT) that organizes the resources using hierarchical model but this time as the AVL tree. AVL tree is a self balanced tree structure that is balanced regarding the height when the nodes added dynamically to the tree which is described by Horowitz *et al.* in [23]. With this method authors argue they provide more scalability and flexibility for grid resource discovery. The RCT using resource characteristics that are usually needed to know to be able to satisfy the queries. The important characteristics that represent the resources are called primary attributes (PA) in [22]. These attributes are organized as the tree nodes to be able to answer the queries based on the ranges. A query is called a range query when it searches the resources of a specific range on a primary attribute (for example available memories > 1 GB). A query that required multiple attributes is called a multi-attribute query.

The RCT proposed by Sun *et al.* [22] is simulated by using Java programming language and compared with hierarchical structure. The simulated RCT and hierarchical model consist of 100 nodes. The query load per node and average search length (ASL) are performance metrics for this simulation. The RCT is found to be 50% more efficient in terms of the number of queries reaching to each node than the Hierarchical model. The average search length is the average number of nodes that a request is passed for query processing. Regarding the comparison of average search length the result shows that RCT searches between 8.5 to 9 nodes and the hierarchical scheme searches all 100 nodes. It is concluded that the Resource Category Tree is an efficient and complete solution for resource discovery.

2.2.4. Grid Resource Discovery Based on Semantic Information

A Peer-to-Peer (P2P) network consists of computers that share their resources with each other without using a server computer. The Peer-to-Peer grid model is similar to the P2P network that consists of a set of super nodes also called grid peers as described by Xiong *et al.* in [24]. A super node or grid peer manages a set of nodes. A node is a computer managing a group of local grid resources. When a user searches for resources, the super node or grid peer domain of local resources is queried first. If no query result is found from a local grid peer, the local grid peer directs the query to the closest grid peers using the random-walks-based method. The P2P system consists of autonomous agents that are able to accomplish unsupervised actions. These are the Request Agent and the Broker Agent. The Request Agent is responsible for the query and result from a grid peer. If it cannot find the information, it sends a request for information to the

Broker Agent of the same grid peer. The Broker Agent contains a database of resource information to be discovered from other grid peers, called the Global Knowledge Database.

As mentioned by Seyed *et al.* [25] a grid computing architecture is required to discover the resources because most of the time of the grid computing is spent on resource discovery. This time can be minimized by adding the architectural layers. These layers will manage all resources of the grid. Resource discovery is becoming more complex because the number of nodes and users are increasing day by day in grids such as Universal Grid.

The Galaxy architecture model proposed by Seyed *et al.* [25] is the hypothetical semantic-based grid architectural model, with the idea of creating a universal grid. It consists of four levels of grid computing in which different forms of indexing called metadata and meta-metadata are employed for resource discovery. These metadata, together with agent programs, are distributed in all layers of the grid. The metadata in Galaxy contains semantic information about the availability of resources that are updated by smart agent programs. In Galaxy, it is assumed that service-oriented programs are available at all levels of the grid and are used by agents when a resource change event is triggered. The metadata contains the latest resource information in XML based format. The important part of Galaxy is the agent and software services, which update semantic information and assist with resource discovery. The semantic information includes ontology (i.e. the meaning and interpretation of information) that is needed for communication between different services. In Galaxy, the administrative information of the resources of any node is created and updated by agent programs and kept in metadata. The Galaxy layer for resource management is on top of the data link layer and under the network layer that has to add additional information into IP packets. The location of the Galaxy layer for resource management is shown in Figure 2.2 on next page.

The use of web services with semantic information, multi-agent systems and metadata as declared by Seyed *et al.* [25] will provide up-to-date resource information to the end user. Although no simulation work is done, it is predicted that resource access will be much faster using Galaxy architecture.

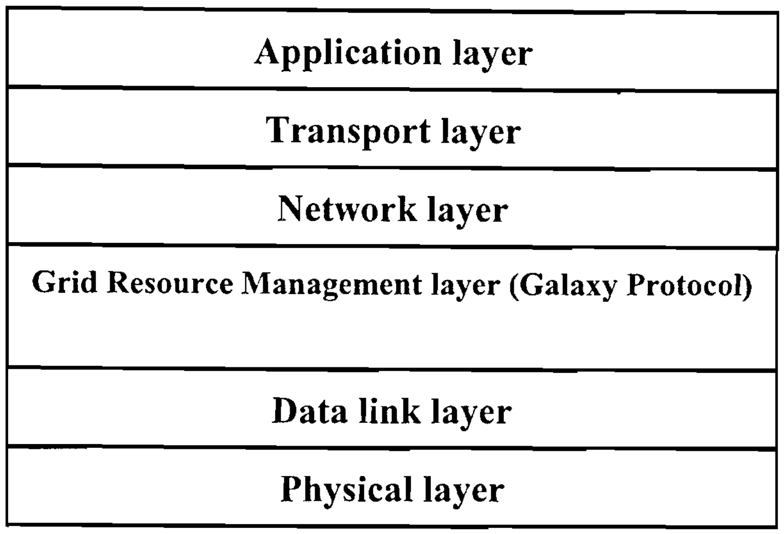


Figure 2.2 Galaxy Protocol layer from [25]

2.2.5. UDDI-Based Resource Discovery

As described by Foster *et al.* [12], the Open Grid Service Architecture (OGSA) is the standard for integration of web services and grid computing. It is introducing the concept of grid services and all the mechanism required for using grid services such as naming, grid service description and grid service discovery. OGSA is the result of Globus toolkit that was introduced by Foster *et al.* in [26]. The Globus toolkit is developed to use the grid services in practice. OGSA is evolving middleware that uses specifications of Web Service Resource Framework to build a web service-based grid.

Benson *et al.* [13] focusing on internal structure of Universal Description, Discovery and Integration (UDDI) for discovering grid services. UDDI is used for resource discovery mechanism of OGSA based grid for grid services. However using UDDI for grid service has some problems because it is designed to be used for business services.

According to Benson *et al.* [13] UDDI registry keeps track of a resource through a string reference key. The UDDI has three basic design issues as the following that make it difficult to be used for grid services.

1. Missing of explicit data type for UDDI directory.
2. Difficulties in handling regularly updating dynamic information such as continuous numeric type of CPU load which changes at instances.
3. The limited query capability.

Benson *et al.* [13] proposed a new UDDI centralized model of grid resource discovery with following modifications:

1. The issue of explicit data type, which does not exist in UDDI registry, is resolved by proposing the continuous variables of numeric type in UDDI registry.
2. The issue of dynamic information is resolved by introducing a new variable called lastUpdateTime in the UDDI registry for storing periodic update from resource providers.
3. The issue of limited query model is resolved by associating performance data like CPU load or machine attributes with a reference key.

Therefore UDDI can be used for grid service discovery but with the above modifications. In this work the experiments are done to find out the performance of modified jUDDI under the system load measured by the update frequency of grid resources. It is mentioned by Benson *et al.* [13] that the implementation of UDDI version 3 will match the requirement for grid services.

2.3. GRID SIMULATING PACKAGES

As mentioned by Buyya *et al.* [27], there may be individuals or organizations who own different grid resources. Each of them might have their own resource management, access and cost policies in the grid environment that makes resource management a complicated issue.

Resource discovery policy is part of the resource management and is also a complicated task. In the future, the number of resources and users in a grid may grow rapidly. Therefore, a grid resource discovery model should meet the scalability issue. To evaluate a scalable grid resource discovery model, we need a grid simulating platform. In the next paragraph we have mentioned the names and brief explanation of the functions of some the grid platforms or simulation packages that can be used for grid simulation.

There are many implemented grids such as DAS3, Grid'5000 as mentioned by Cappello *et al.* in [28]. There are also simulation packages to simulate a grid environment. For example the Bricks defined by Takefusa *et al.* [29] simulates the behavior of resource scheduling algorithms for global networks. The SimGrid elaborated by Casanova *et al.* [30] is a grid simulation toolkit to simulate scheduling of distributed and parallel applications on network and distributed computing platforms. The GridSim explained by Buyya *et al.* [31] is a simulation toolkit for modeling and simulation of grid entities. These entities may be resources, applications, users and resource brokers/schedulers in parallel and distributed computing systems. The GangSim described by Dumitrescu *et al.* [32] is a tool developed for grid scheduling studies and control of resource sharing. The OptorSim described by Bell *et al.* [16] is a grid simulator toolkit that is developed to examine different replication approach. The WGridSP elaborated by Kang *et al.* [33] is a grid scheduling toolkit for stimulation in a Java environment that performance evaluation of a grid in Internet. The WGridSP is based on GridSim.

There are many types of simulation packages as mentioned in the previous paragraph to simulate different resource discovery models. The commercial success of grid computing depends on the right choice of resource discovery model. Therefore we need to select a simulating package that would be used to evaluate the scalability of our grid resource discovery architectural model.

2.4. COMPARISON OF OUR WORK WITH OTHER GRID ARCHITECTURES FOR RESOURCE DISCOVERY

Our work is based upon web services and the simulation package named GridSim that is simulating a layered architecture for grid resource discovery. The GridSim simulates resources, users, jobs and etc. The GridSim is mentioned in section 2.3 and will be discussed more in Chapter 4. Comparison of our work and similar works are given next.

Regarding to the use of central approach of discovering web services for grid computing, our work is similar to that is presented by Benson *et al.* [13]. We use UDDI but we don't focus on the internal structure of UDDIs for discovering grid services. However we assume a required UDDI technology exists according to that proposed by Benson *et al.* [13]. Therefore we are able to use UDDIs for discovering resource finder web services. We also assume the resource finder web service proposed by our work has the requirements of a grid service that suggested by Foster *et al.* [12] for OGSA but we do not focus on its internal structure.

Our proposed model is the simplification of upper two layers of the layered architecture model discussed in section 2.2.1. This simplification is done only for the application and collective layers of the layered architectural grid computing presented by Foster *et al.* [20]. The application layer of our model provides web interface for the user and the collective layer of our model is a web service focusing on resource discovery. They are therefore referred to as Web Interface and Resource Finder Web Service layers respectively, as shown in Figure 3.1. We do not address the rest of the three lowest layers of the layered architectural model presented by Foster *et al.* [20], but a combination of all of them is shown as the Grid Computing Resources/Nodes layer in Figure 3.1. The Web Service in our model receives the client's request for a resource and finds an appropriate resource finder that returns a list of requested resources and their contact addresses. The Middleware is used for communicating between the layers of the model. Metadata shown in Figure 3.1 is the information on the resources that is handled by the resource finder. Metadata is in XML format because it will be sent to the requesting client through the web. Using web services and XML-based metadata for grid computing are also proposed by Seyed *et al.* [25] in Galaxy architecture that is outlined below.

Similar to Galaxy, our model uses web services, but they have been used for web interface and resource finder layers. We do not assume web services in the lower layers of the grid exist. We also use metadata in XML format that contains the information of the resources. However, in our proposed architecture, we assumed metadata would be maintained and updated by a resource finder through the information passed on by regional/local managers, grid resource providers or other event-based programs. In our model we did not focus on the required mechanisms in the lower layers required for updating metadata. Moreover, our proposed architecture does not use any semantic layers or agents that are required at all levels of the grid, as in the Galaxy model.

Another major difference between implementation of our model and implementation of the Galaxy model is considering TCP/IP stack protocols. The location of the Galaxy layer (for grid resource management) is suggested to be placed on top of the Data Link Layer and under the Network layer as described by Seyed *et al.* [25]. This means that to implement Galaxy after the IP protocol assembled the packets, the information generated by the Galaxy layer should be added to each packet. However, considering the implementation of our model regarding TCP/IP, our model can be implemented in a way that is similar to the DNS protocol presented by Giordano [34]. The additional layer for our proposed model could be placed in the application layer as Web Interface, and the SOAP messages would transfer between the resource finder and client web services. There can be many resource finder repositories in our model similar to Name Servers in DNS that can also be used for sending queries to other resource finder repositories to find requested resources.

We believe that by limiting the web services for communication between user and resource finder components and simplifying the metadata, our model can be implemented on top of an application layer of TCP/IP. Therefore, we have solved the main drawback of Galaxy that is impracticality. In implementation of the Galaxy model not only it is required to change the lower layers of TCP/IP protocol, but it is also assumed that software services are available in all layers of grid computing platforms.

CHAPTER 3

3. PROPOSED MODEL AND IMPLEMENTING STRATEGY

This chapter discusses our proposed model. The discussion about the proposed web-based architecture will be presented in section 3.1. This chapter also elaborates on the implementation strategy for our proposed web-based layered architectures in section 3.2. Finally conclusions are presented in section 3.3.

Section 3.2.1 explains the implementation of a Direct Web-Based Grid Resource Discovery model. Section 3.2.2 describes the employment of jUDDI registry for implementing a Centralized Web-Based Grid Resource Discovery model.

3.1. PROPOSED WEB-BASED ARCHITECTURE FOR GRID RESOURCE DISCOVERY

Resource discovery is a time-consuming part of grid computing. To implement more effective resource discovery, we propose a model to employ web services on the upper layers of grid computing. Although our model may be considered to be the simplification of some of the previous proposed models presented by Foster *et al.* [20] and Seyed *et al.* [25], the structure and function of our model differ significantly from these models. This is explained in further detail in this section. The structure and function of our proposed model is shown in Figure 3.1 on the next page.

In our proposed model, a user will access web services through the highest layer of the grid to locate the resources with the help of a resource finder (or resource broker) and metadata. The web services will then contact the resource finder to obtain the address of a resource. If the requested resource(s) are available within the metadata, the resource finder will send the metadata related to that resource through the web service to the user. Metadata is the highest level of resource information, including addresses, in the grid. To discover the resource finder service on the Internet, the client can send queries to repositories similar to Name Servers in Domain Name Service (DNS) protocol [7] as proposed by Giordano [34]. Metadata contains the highest level of information of the group of resources available within the local network, or within the same region containing special criteria. The user who requests the resource may know the URL of the resource finder (for example by using a search engine or being in the same region) or may be redirected to send DNS queries to the repositories (UDDI) of the resource finder services to find the URL of the related resource finder. Once the URL of the resource finder that has that resource/s on its list has been found, it can be accessed by the user through a related web service. Thus in our model, end users will use web services in both of the above mentioned cases only to communicate with a resource finder that has the list of the group of the requested resources in its metadata.

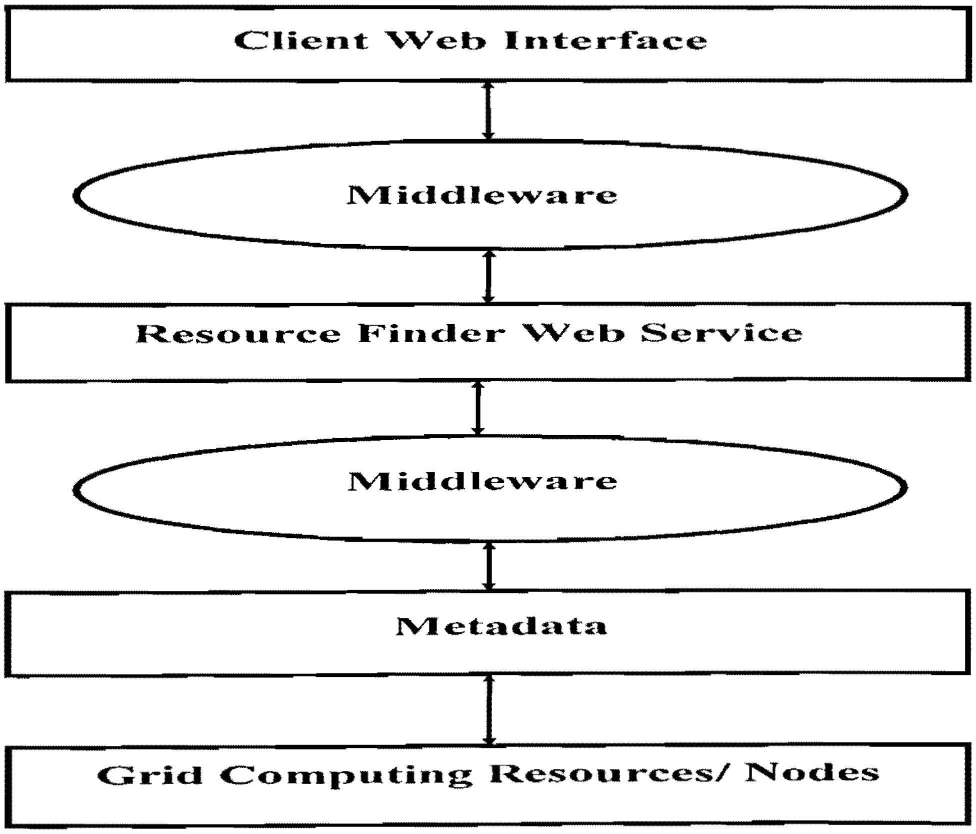


Figure 3.1 Proposed Model

We have submitted paper discussing the implementation of the proposed model [35]. The modified layers in our model are client web interface and resource finder web services. The resource finder service searches metadata and returns a list of the available resources of the grid to the user by using common protocols within the application level of TCP/IP such as HTTP. Our model proposes XML format for metadata to be used by web services. As mentioned in section 2.4, our proposed model could be considered a simplified form of application and collective layers of grid Layered Architecture presented in [20]. Similar to the Galaxy model presented by Seyed *et al.* [25], our model uses web services to find grid resources. However, our model does not use any semantic components or smart agents because we are concern with practicality and simplicity of the proposed model.

The search for the resource finder and metadata components of our proposed model can be considered similar to the distributed search model in Name Service (DNS) protocol presented by Mockapetris [7], with slight modification that is explained next.

By using a web interface, the user sends a request to access the grid resources from a regional resource finder web service. The resource finder searches its metadata that contains the latest information on the resources, including their location in the grid. We assumed the information on grid resources that can be provided by grid resource providers, local regional managers or special programs can be fed to metadata and updated periodically. The simplest scenario is a regional resource finder that is known to the user in advance that has the resources requested by the user and acts as the broker to allocate these resources within the grid network. However, there are the situations that are not that simple, and they are explained next.

First, we consider a scenario in which the resource finder realizes there are many resources eligible to satisfy users search criteria. In that case a resource finder communicates with the user through a client web interface to ask the user to select the required resources from the list. Second, we consider a scenario where the resource finder, by searching metadata, realizes there is no resource or a low number of resources available for the user. In that case, the user's search query will be sent to a regional registry service that contains the list of resource finders and high level information on their metadata. The search for finding a resource finder that has the resources required by the client may continue in a similar manner to the distributed search model performed by name servers for mapping URLs to IP addresses in DNS protocol. The final result of this search in our model is the URL of the appropriate resource finder service that has the requested resources. The result of this search will be returned to the client. This approach described by Giordano [34] and suggested in our model is called "a distributed search".

Since resource finders are web services, we assumed that similar to the approach proposed by Giordano [34], DNS queries are used to discover the URL of the appropriate resource finder service by contacting central registries of each region in which web services are registered. This is a traditional method of using UDDI to find web services. The UDDI registries and the high-level information of the resource finder services, and their metadata are similar to name servers and Zone files, respectively in the DNS protocol.

We used the distributed search method in the proposed model not only because of the scalability of this approach, but also because this method is used in the model to update the information of resource finders registries based on the latest information on the grid. When the distributed search for resource finders is in progress, each resource finder registry in the path finally receives the information of the appropriate resource finder that has the requested resources. Each of these resource finder registries in the path adds this new information to its list of registered services, similar to the DNS name servers when they cached the recently resolved URL and IP address. Thus by employing this approach, not only do resource finder's metadata contain the latest information of the resources in their own region, but the resource finder registries in the long run can also cache the recent information of other registered resource finders and their resources.

After finding the URL of the appropriate resource finder that has the list of required resources, the resource finder performs the duties of a resource broker by allocating the resources within the grid. It will submit the jobs to these resources, and after a job executes, it will deliver the results to the user.

3.2. UNDERLYING TECHNOLOGY FOR IMPLEMENTATION OF THE PROPOSED ARCHITECTURE

Our main goal throughout the implementation of the model is to develop a prototype that can be used together with grid simulator software to estimate any changes in grid resource discovery time as a result of adding our model to the top of the grid. For implementation of the resource finder, we used a web service. For a client web interface that calls for a resource finder service by the user, we implemented a web application. For simplicity we assumed that the scenarios given next are within the regional search for a resource finder service. Because of the complexity of a DNS-based search, we have not implemented the whole model of a distributed search proposed by our model in this work. The implementation for two scenarios of regional search is as follows:

1. First, we implemented a model in which the resource finder service end point (URL) is known to the client and the resource finder has all the resources. Since in this approach there is no need to discover a resource finder service, and a client can call the resource finder service directly, this approach is called Direct Web-Based Grid Resource Discovery Model (DWGRD). The implementation of DWGRD is elaborated upon in section 3.2.1.

2. In the second implementation approach, the resource finder web services are registered in the regional central registry, which is a part of our web application. In this approach the user knows the address of a registry location such as UDDI, by which he or she can find the URL of resource finder service that has the required resources. The discovery of a proper resource finder by using UDDI is added to the DWGRD model and the resulting architecture, is called the Centralized Web-Based Grid Resource Discovery (CWGRD) model. Once the service is found, the client then sends the request of requirements to the service, and the result is returned to the client. CWGRD is discussed in section 3.2.2.

3.2.1. Direct Web Service-Based Grid Resource Discovery Implementation

For implementation of the DWGRD model, NetBeans IDE version 6.5.1 is used because of its strength in the implementation of web service applications [36]. The resource finder web service is accessed by a web application called client web application. The resource finder web service calls GridSim to simulate accessing the resources in the grid environment.

The messages are then exchanged between the web service and client web application using Simple Object Access Protocol (SOAP). The web service and client web application are executed on the same server. The implementation flow of Direct Web-Based Grid Resource Discovery model in NetBeans IDE 6.5.1 is shown in Figure 3.2 and simulation results are given in Chapter 4.

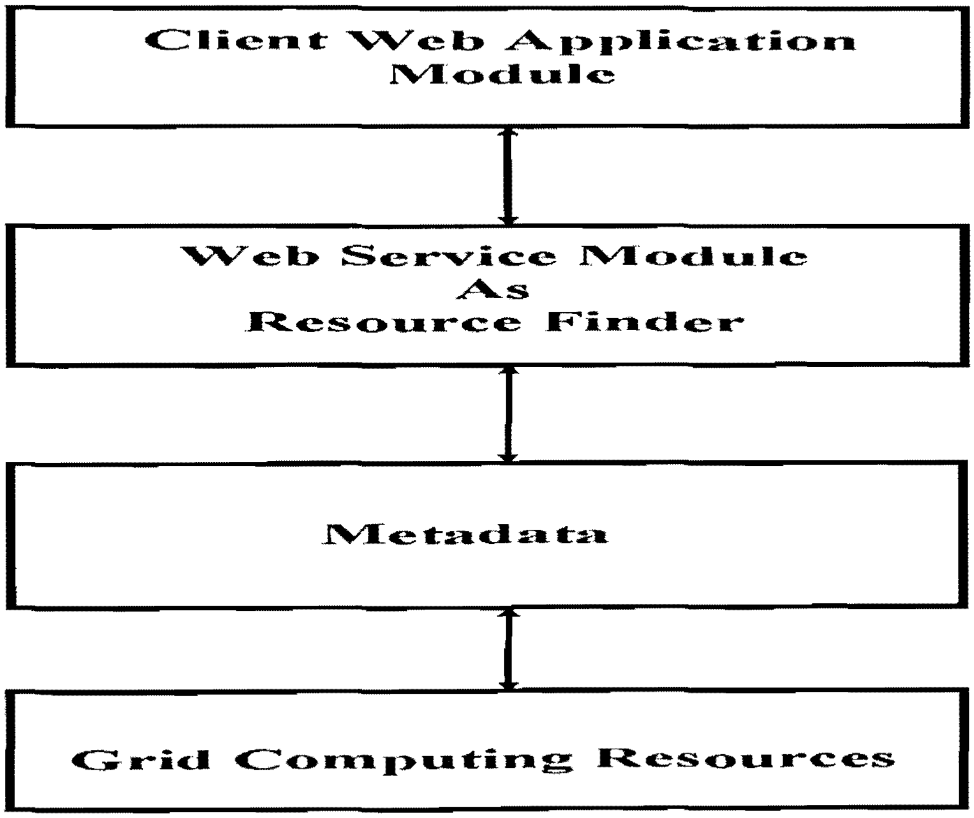

Client Web Application Module
Web Service Module As Resource Finder
Metadata
Grid Computing Resources

Figure 3.2 Implementation Flow of DWGRD

The implemented software has two parts:

1. The client web application has a JSP page as an interface that sends user's entered parameters to the web service application. The web service module acts as a grid resource finder. The client web application receives timing results from the grid resource finder after the next step.

2. The grid resource finder web service calls the Grid Information Service class of GridSim, which simulates grid entities such as users, resources and jobs in the grid environment.

3.2.2. Centralized Web Service-Based Grid Resource Discovery Implementation

The CWGRD model is also implemented as a web application in NetBeans IDE Version 6.5.1. The UDDI registry technology used is called jUDDI. The detailed explanation of installation of jUDDI in Apache Tomcat Server is given in [37] and specific implementing detail is given in Appendix A.

We are using UDDI and jUDDI registries while assuming, they have been enhanced by the solution provided in [13] and are capable of discovering grid resources. The jUUDI server is used for our model for the following reasons:

1. It keeps the information in XML format and it compatible with web service technology.
2. It keeps all the details required for discovering resource finder web services.
3. We use multiple UDDIs distributed in Internet instead on the central UDDI. It means we will use multiple jUDDI servers at many regional bases similar to DNS servers. These jUDDI servers will be communicating with each other using distributed search for resource discovery similar to DNS servers. Using multiple jUDDI servers, we will not have the problem of a central bottleneck when having only one repository.

Once the service is registered into the jUDDI database that is connected to the databases server, the service can be accessed from the registry using web application. The CWGRD model is implemented as shown in Figure 3.3.

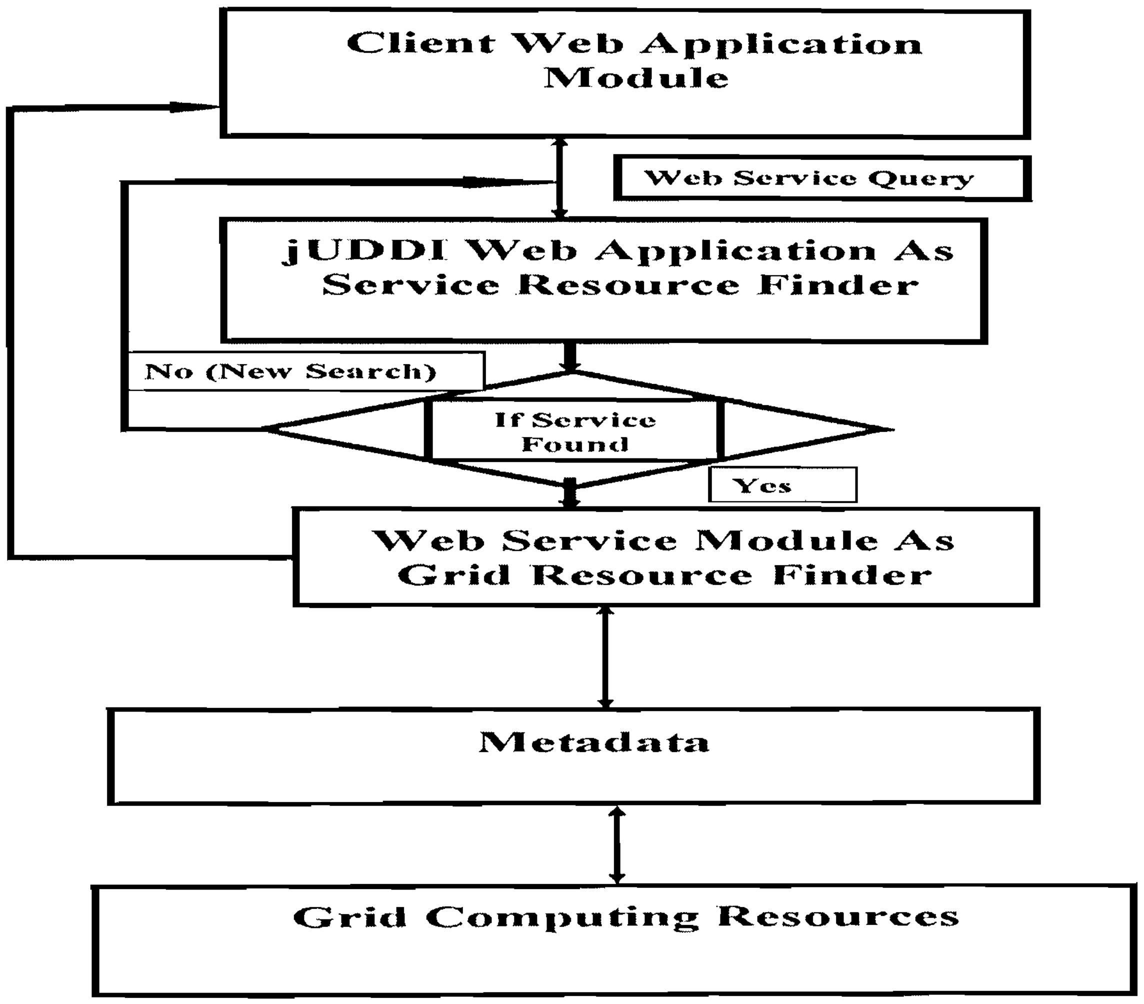

Client Web Application Module
Web Service Query
jUDDI Web Application As Service Resource Finder
No (New Search)
If Service Found
Yes
Web Service Module As Grid Resource Finder
Metadata
Grid Computing Resources

Figure 3.3 Implementation Flow of CWGRD

A client web application is constructed in such a way that it sends the inquiry to find service from the jUDDI registry. The result of the service query is returned and the time in milliseconds that it took to satisfying the query from the jUDDI registry is recorded. This time will be added to the time reported by DWGRD (details will be explained in the next chapter) that is determined by the resource finder service.

3.3. CONCLUSIONS

In this Chapter we have proposed a web service-based layered architecture that is a scalable solution for resource discovery in grid computing. This model uses DNS-like queries to find the URLs of required services by contacting the central repositories that are distributed throughout the Internet. The underlying technologies for implementation of the architectures according to two common scenarios of this model are introduced.

CHAPTER 4

4. SIMULATION AND RESULTS

Many grid simulation toolkits have been developed for simulation of grid environment. Buyya *et al.* [31] presented the discrete-event-based grid simulation toolkit designed in java called GridSim. The GridSim toolkit is designed to support modeling and simulation of grid entities. The heterogeneous resource(s) (both time- and space-shared), user(s), resource broker(s) and application(s) are generated using GridSim as grid entities. The GridSim simulation package is described in section 4.1.

Section 4.2 discusses the simulation results of the GridSim (i.e. no architecture) and Direct Web-Based Grid Resource Discovery (DWGRD) model. Section 4.3 discusses the simulation results of the Centralized Web-Based Grid Resource Discovery (CWGRD) model. The discussion of scalability is presented by comparing GridSim, DWGRD and CWGRD under increasing the grid load is in section 4.4. The statistical analysis of the results obtained from this test that we called scalability test is presented in section 4.5. The conclusions are given in section 4.6.

4.1. GRID SIMULATION

The major challenge of this work was finding a grid simulator toolkit suitable for our work. Firstly, grid simulation is different from network simulation and, secondly, we needed to find a simulator capable of estimating resource discovery time. GridSim is a java-based discrete event-based simulator that we found suitable for our work.

GridSim simulates entities in the heterogeneous grid environment. These entities are user(s), application(s), resource(s) and resource brokers/schedulers in parallel and distributed computing systems as described by Buyya *et al.* [31]. The GridSim v4.1 consists of 161 classes, among them Grid Information Service, that is used for resource discovery in the grid, indexing and registration. Availability of this class that can be used for our grid resource discovery simulation plus modular architecture for modeling a layered based grid are the main reasons that we selected GridSim for our simulation.

Within the simulation for each of the architecture, the grid simulation is triggered by the resource finder web service of our model. There is a call to GridSim.init() that initialized the object of Grid Information Service (GIS) class, that as explained before is used for grid resource management simulation. We averaged the round trip time of a packet reported by GridSim for all users and used it as the estimation of the time that is required for grid resource discovery by a user in our experiments.

Since GridSim is a discrete event-based simulator, as suggested by Law *et al.* [38] we performed the complete standalone simulation test for all three models to make sure that the achieved results have steady-state behavior. It means we performed simulation for all the combination of resources and users (i.e., 36 points for each model) for the three models (explained in section 4.2) to make sure the achieved resource discovery times are non-transient and stable. After that we compared the results of the models for the selected points to be able to test the scalability of the system.

4.2. SIMULATION OF DIRECT WEB SERVICE-BASED GRID RESOURCE DISCOVERY

For both DWGRD and CWGRD approaches, the resource finder service calls and executes GridSim (that is run on a same server) which is responsible for generation of grid entities such as resources, GIS nodes and users. GIS node is a managing computer that registers and keeps the information of resources within its region.

To implement both DWGRD and CWGRD, the one client and one server architecture is used. The reason is that we simulate a grid on one machine. Therefore, for providing comparable timing results with GirdSim, we ignored the network latency involved in calling the resource finder services in DWGRD or communicating with UDDI in CWGRD. For real systems, this architecture can be extended to multiple clients with multiple servers, when the resources of the grid are located all over the grid network.

The resource discovery times of GridSim (for simulating the grid network with no architecture on top), Direct Web-Based Grid Resource Discovery (DWGRD) and Centralized Web-Based Grid Resource Discovery (CWGRD) models are produced by simulation.

The DWGRD model is developed and executed in NetBeans IDE 6.5.1 as explained before in section 3.2.1. The client web service application communicates with the relevant resource finder web service application. The grid resource finder web service and client web service applications are deployed and executed many times on an Apache Tomcat version 5.5.23 server. The number of grid users and number of resources are simulation parameters that are passed to GridSim to observe the difference of resource discovery by using the Direct Web-Based Grid Resource Discovery model as compared to no architecture (i.e., when running only GridSim).

The number of resources is used as a first variable parameter in simulation with increasing numbers of users as the second parameter, while the number of GIS nodes is kept constant throughout simulation to 2. The Direct Web-Based Grid Resource Discovery model is for the first scenario that is added on top of GridSim. The proposed model and GridSim (no architecture) both are executed for ten times for each pair of the number of users and resources. First we did a complete test for GridSim as the following:

The average resource discovery time of a user for GridSim (no Architecture) is computed after ten times of data collections for each number (1, 20, 40, 60, 80 and 100) of users while keeping the number of resources constant to 1, 20, 40, 60, 80 and 100. This process was repeated for the constant users and variable resources for the same numbers. The complete graph is shown in Figure 4.1 below. The resource discovery time for each user increases consistently as the number of users and resources are increased in GridSim. The resource discovery time for each user increases as the number of users and resources are increased because GridSim simulates more users and resources. The average time of resource discovery for an average of 100 users with 100 resources in GridSim is found to be 12.012 seconds.

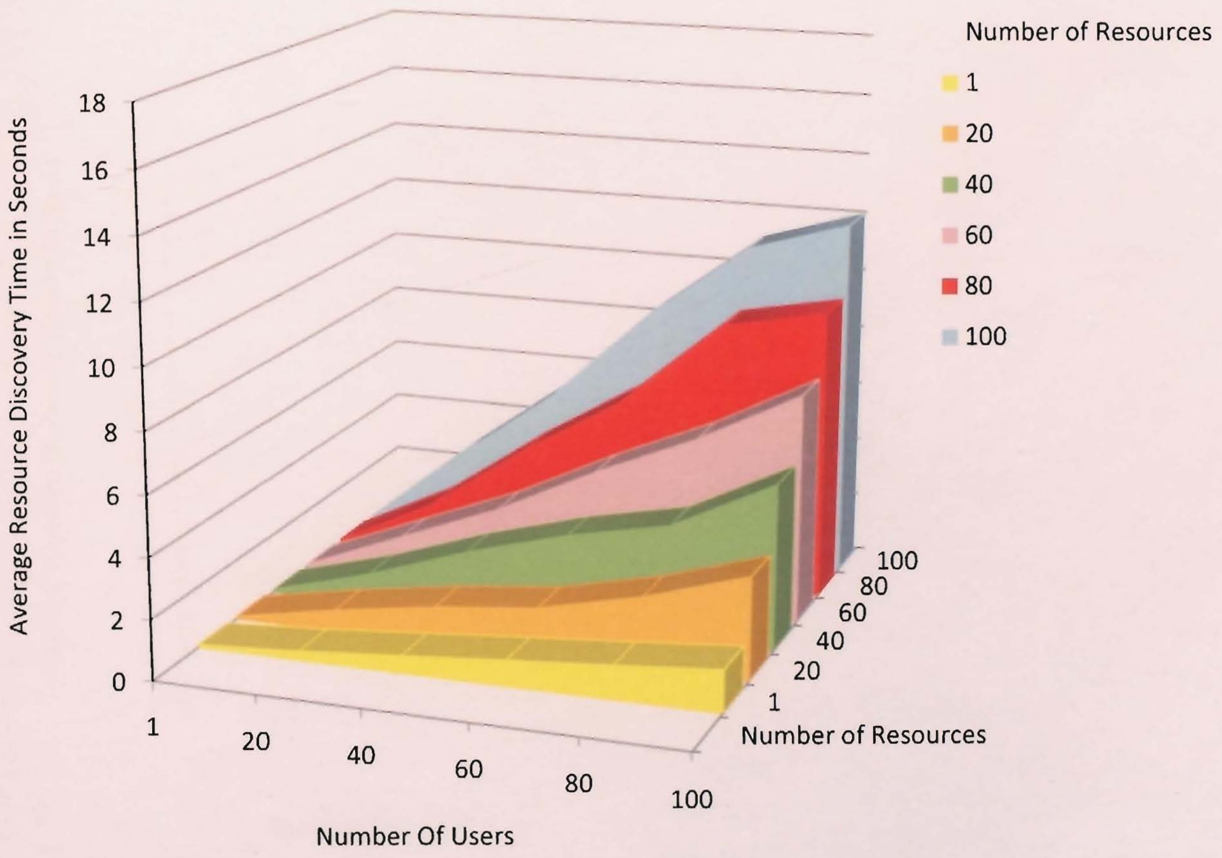


Figure 4.1 Average User Resource Discovery Time of GridSim (i.e. no architecture)

Now we perform a complete test (i.e., the combination of resources and users) for DWGRD as the following:

The resource discovery time of the Direct Web-Based Grid Resource Discovery model is the resource discovery time reported by GridSim when is triggered by the resource finder web service plus the time of calling the web service. The data for each user of the Direct Web-Based Grid Resource Discovery model is also collected ten times for each number (1, 20, 40, 60, 80 and 100) of users while keeping the number of resources constant to 1, 20, 40, 60, 80 and 100 and vice versa similar to the method used for GridSim as shown in Figure 4.2 below.

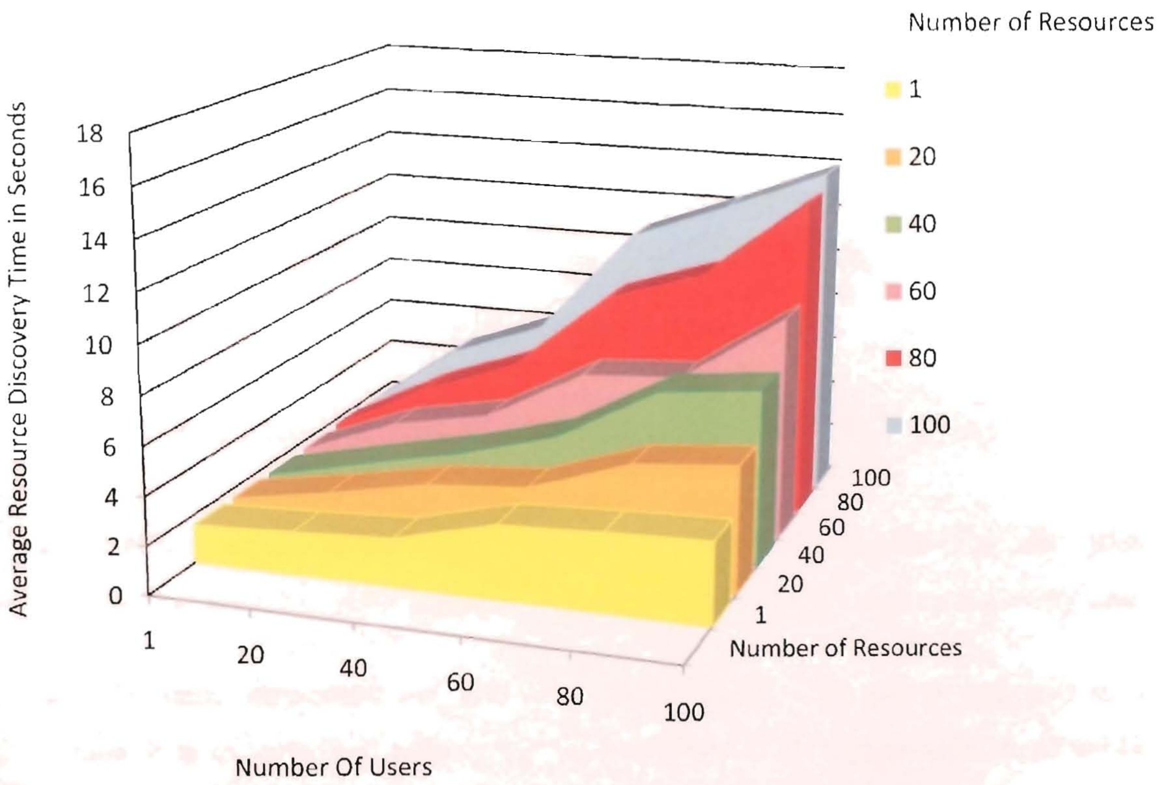


Figure 4.2 Average User Time of Direct Web Based Grid Resource Discovery (DWGRD)

The resource discovery time for each user also increases slowly as the number of users and resources are increased in the Direct Web-Based Grid Resource Discovery model, as shown in Figure 4.2. The resource discovery time for each user increases as the number of users and resources are increased because GridSim simulates more users and resources. The average time of resource discovery for an average of 100 users with 100 resources in the Direct Web-Based Grid Resources Discovery model is found to be 13.902 seconds.

The average user times of the GridSim and Direct Web-Based Resource Discovery model will be compared in section 4.4. The slightly higher average time for resource discovery by increasing the number of users and resources in the Direct Web-Based Grid Resource Discovery model indicates the additional time for using web services to send the parameters to GridSim.

4.3. SIMULATION OF CENTRALIZED WEB SERVICE-BASED GRID RESOURCE DISCOVERY

The Centralized Web-Based Grid Resource Discovery model is also developed and executed in NetBeans IDE 6.5.1 as explained in section 3.2.2. The Centralized Web-Based Grid Resource Discovery model consists of three applications:

1. juddi: A java based UDDI web application that is a centralized registry.

2. GridArchitectureTomcatWebClientjuddi: A client web service that calls juddi to discover the URL of the resource finder and to collect the resource discovery time.

3. The same application that was used for DWGRD to be able to call grid resource finder to execute Grid Information Service (GIS) class of the GridSim, as explained before GIS class generates indexes and registers grid entities.

The service discovery parameters are entered similar to DWGRD and information flow is as shown in Figure 3.3. The timing result together with the discovery time of the URL is returned by the program. The discovery time of the URL is recorded and control is returned to the client web service application, where a web interface is given to enter the URL of the discovered grid resource finder web service. Thereafter the web service application works similar to Direct Web-Based Grid Resource Discovery model as described in the previous section. The Centralized

Web-Based Grid Resource Discovery model is a set of applications that are deployed and executed many times on the Apache Tomcat version 5.5.23 server to record the discovery time of the URL. The discovery time of the URL recorded in this approach is the sum of the jUDDI search time and running time of the application. For each point this time (i.e., resource finder discovery time) was added to the average resource discovery time of the Direct Web-Based Grid Resource Discovery that was computed in Section 4.2 to get the total resource discovery time of the Centralized Web-Based Grid Resource Discovery application (CWGRD) under the increasing number of users and resources. The average resource discovery time for number of users=100, number of GIS node=2 and number of resources=100 of Centralized Web-Based Grid Resource Discovery application (CWGRD) is found to be 15.618 seconds, as shown in Figure 4.3 below.

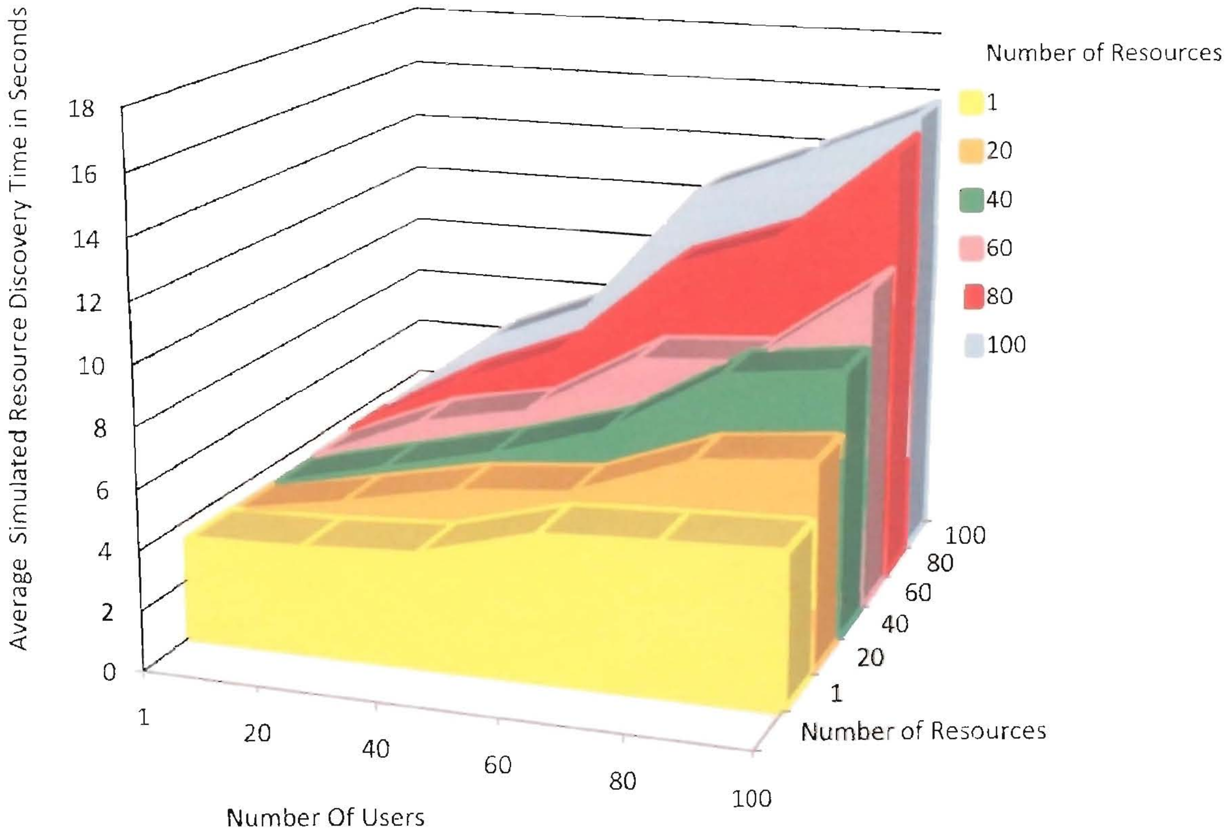


Figure 4.3 Average User Time of Centralized Web Based Grid Resource Discovery (CWGRD)

As mentioned in section 4.2 the average resource discovery time of the Direct Web-Based Grid Resource Discovery application (DWGRD) for the same parameters is found to be 13.902 seconds, and the average resource discovery time of GridSim for same criteria is found to be 12.012 seconds, as given in section 4.2.

The average user time of GridSim, DWGRD and CWGRD will be compared in more detail in section 4.4. The higher average resource discovery time (as shown in Figure 4.3) by increasing the number of users and resources in the Centralized Web-Based Grid Resource Discovery model indicates that this architecture is adding additional time to the Direct Web-Based Grid Resource Discovery application because it searches the URL of web service from a database. By doing complete simulation for the models, we could not find any abnormality in the system and we conclude all models have steady-state behavior.

4.4. DISCUSSION OF SCALABILITY

As expressed in the previous sections, some of the Grid parameters that change resource discovery time are simulated by GridSim. Other factors are part of our model and depend on whether resource discovery is done through direct or centralized architecture. To be able to compare all three models in this part, first we looked at the effect of increasing resources with fixed numbers of users 20, 60, and 100 with all the same conditions as in the above sections. We also used the same data as given in the above sections for each model. The two dimensional graphs that demonstrate the results are shown next in Figures 4.4, 4.5 and 4.6 respectively.

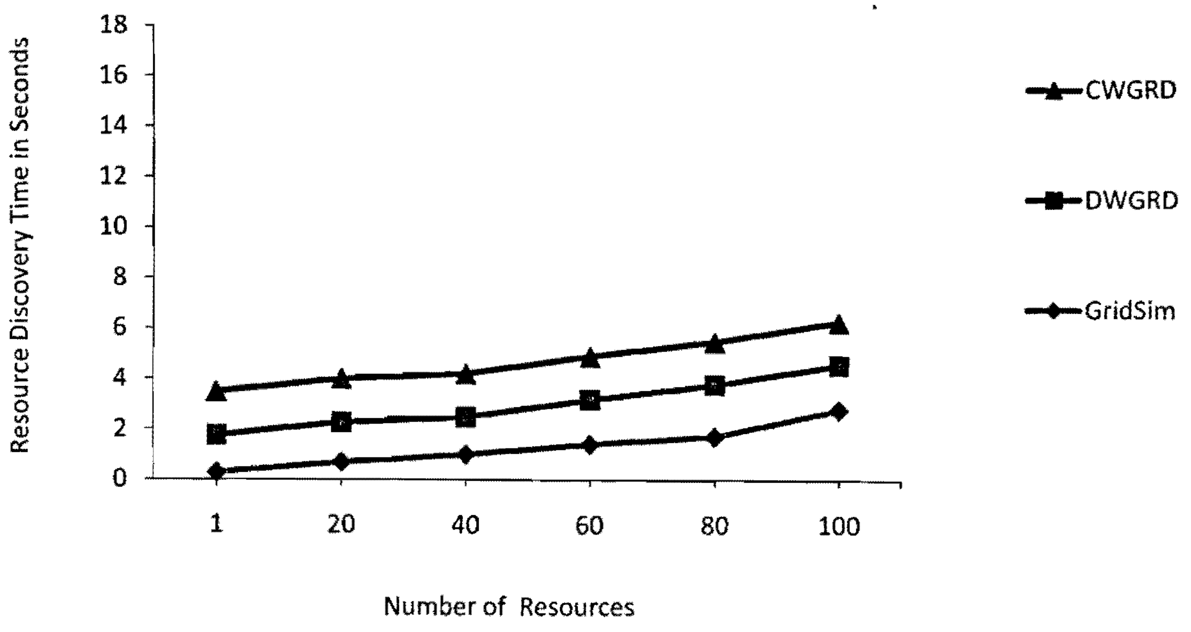


Figure 4.4 Average User Time of GridSim, DWGRD and CWGRD with users=20

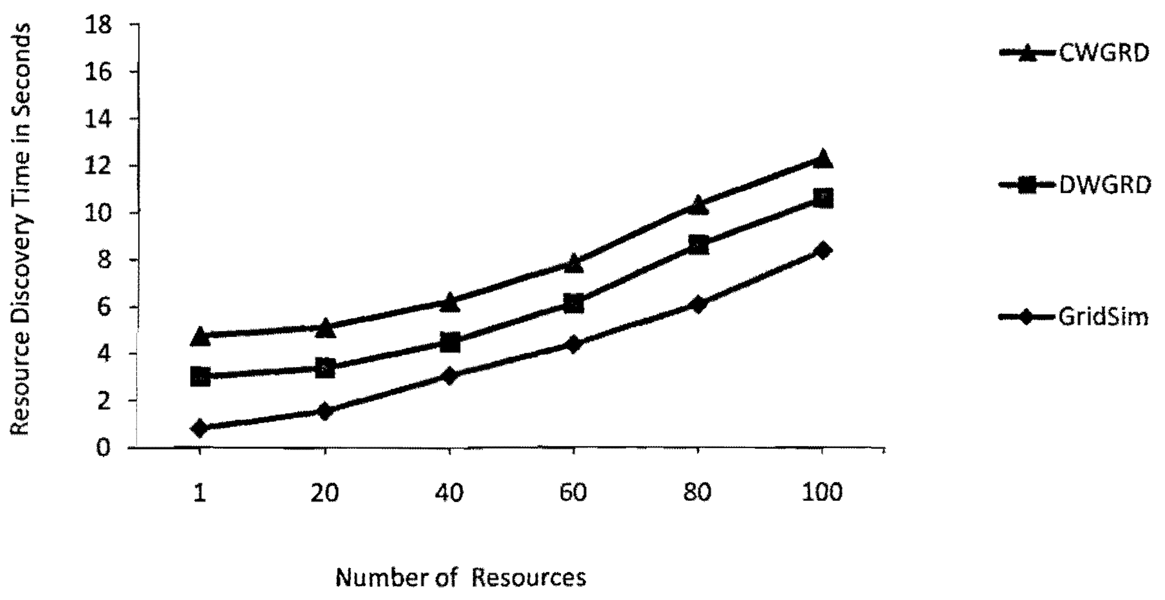


Figure 4.5 Average User Time of GridSim, DWGRD and CWGRD with users=60

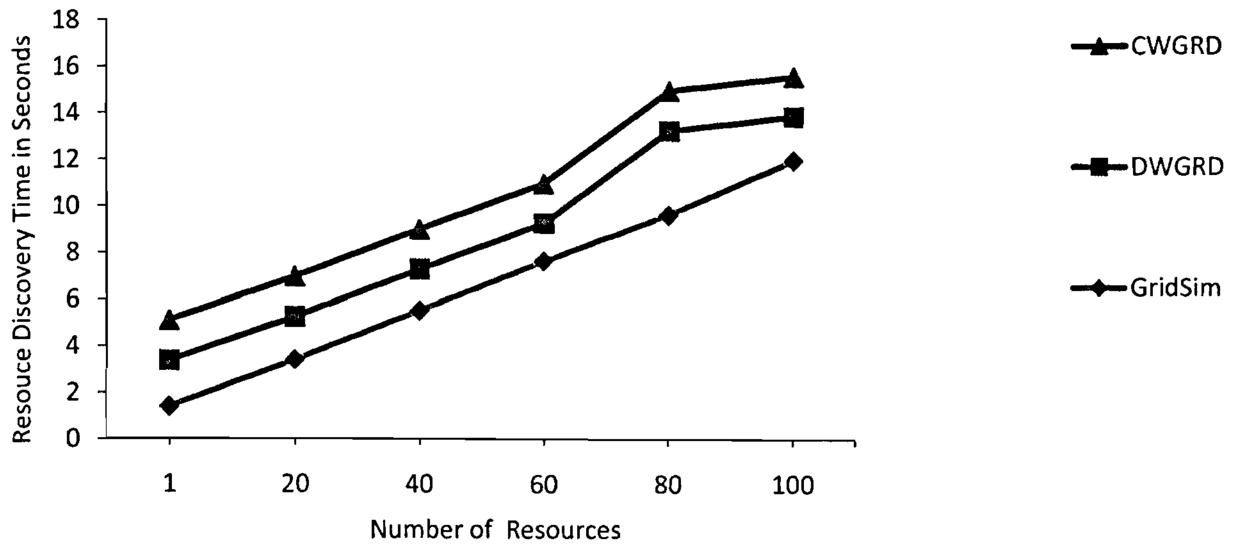


Figure 4.6 Average User Time of GridSim, DWGRD and CWGRD with users=100

The reason for the increase of resource discovery timing is generation of more simulated resources in GridSim (No Architecture), DWGRD and CWGRD. The Direct Web-Based Grid Resource Discovery time includes the web service execution time and the time to discover resources through the resource finder web service plus GridSim resource registering time. The Centralized Web-Based Grid Resource Discovery Architecture time includes the time to discover the URL of web service from jUDDI registry that is the time to search the registry and running the application plus the time of DWGRD that includes the time of calling web service plus GridSim resource registering time.

Since we want to test the scalability, increasing the number of users with the fixed number of resources set to 20, 60 and 100 is the other criteria that we considered to observe for analyzing the effects of increasing users on resource discovery time.

A constant increase in resource discovery timing is found as the number of users is increased with constant resources=20, 60 and 100 in the Centralized Web-Based Grid Resource

Discovery, Direct Web-Based Grid Resource Discovery and GridSim models as shown next in Figures 4.7, 4.8 and 4.9.

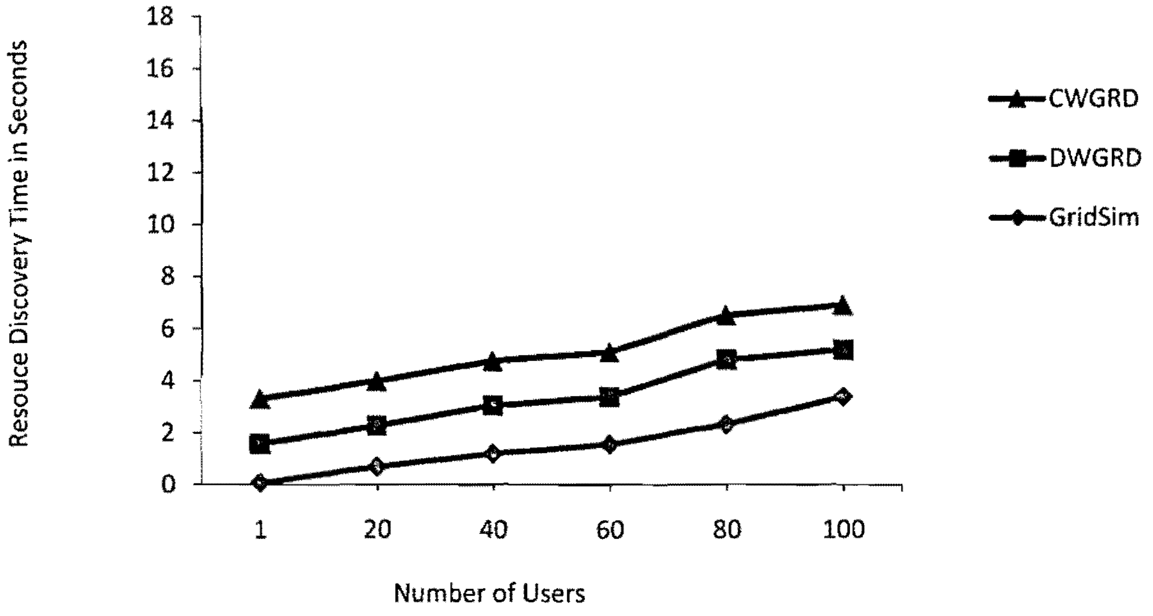


Figure 4.7 Average User Time of GridSim, DWGRD and CWGRD with resources=20

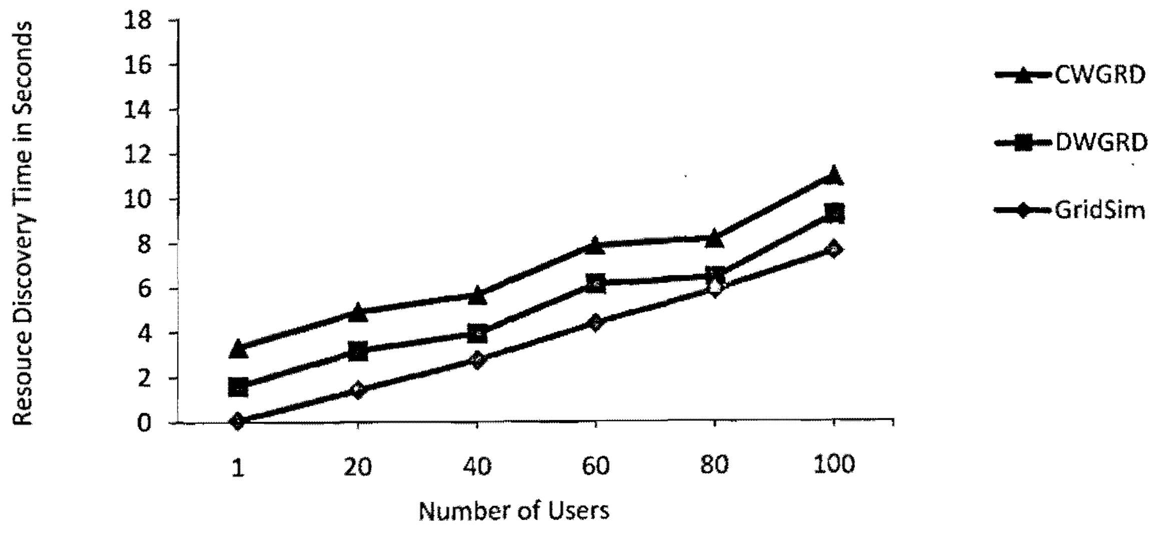


Figure 4.8 Average User Time of GridSim, DWGRD and CWGRD with resources=60

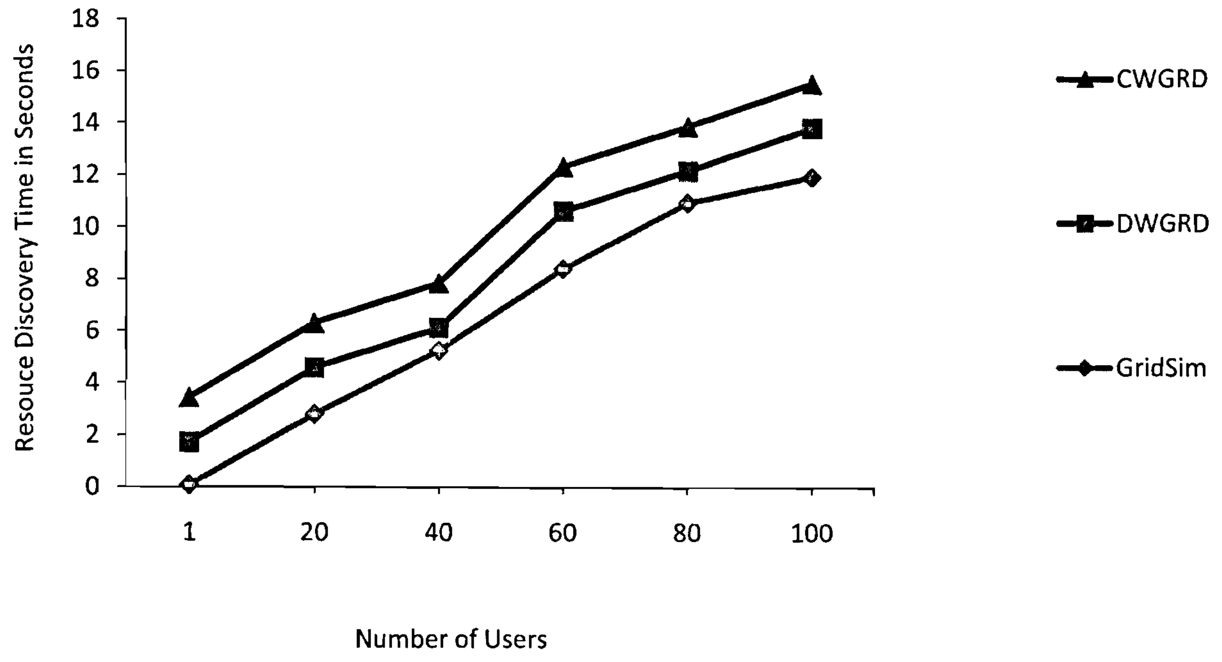


Figure 4.9 Average User Time of GridSim, DWGRD and CWGRD with resource=100

As these graphs show, the average time of resource discovery of an average of 100 users with 100 resources in the Centralized Web-Based Grid Resource Discovery model is found to be 15.618 seconds. The average time of resource discovery of an average of 100 users with 100 resources in Direct Web Based Grid Resource Discovery model is found to be 13.902 seconds, while that of GridSim is found to be 12.012 seconds. As we can see in Figures 4.6 and 4.9, the difference between resource discovery time of GridSim, DWGRD and CWGRD are the same for a fixed 100 users and 100 resources. We also want to see the difference between resource discovery times of three models when increased number of users and resources are equal.

The GridSim simulates grid entities such as users and resources with their registering time. The GridSim (No Architecture) is a simple java application that executes in NetBeans IDE 6.5.1 without any use of web services. It is a base for Direct Web-Based Grid Resource Discovery and Centralized Web-Based Grid Resource Discovery models. The Grid Resource Discovery Time found in GridSim is less than both DWGRD and CWGRD. However when the number of resources and users increased the discovery time of GridSim become closer to the discovery times of two models

The main goal of performing these tests is to test the scalability of the system. We want to see after which number of resources and users the difference between the cost of the models and GridSim become negligible. Although there is a differences between the average resource discovery times of three models but we can observe that this difference decreases under more load of grid. In the other words the mean resource discovery time increases less in DWGRD and CWGRD compared to GridSim. We used bar graphs for the same points 20/20, 40/40, 60/60, 80/80, 100/100 users and resources to observe the increase in means of resource discovery times of DWGRD and GridSim models and CWGRD and GridSim models. These comparisons are presented in the next two graphs in Figures 4.10 and 4.11.

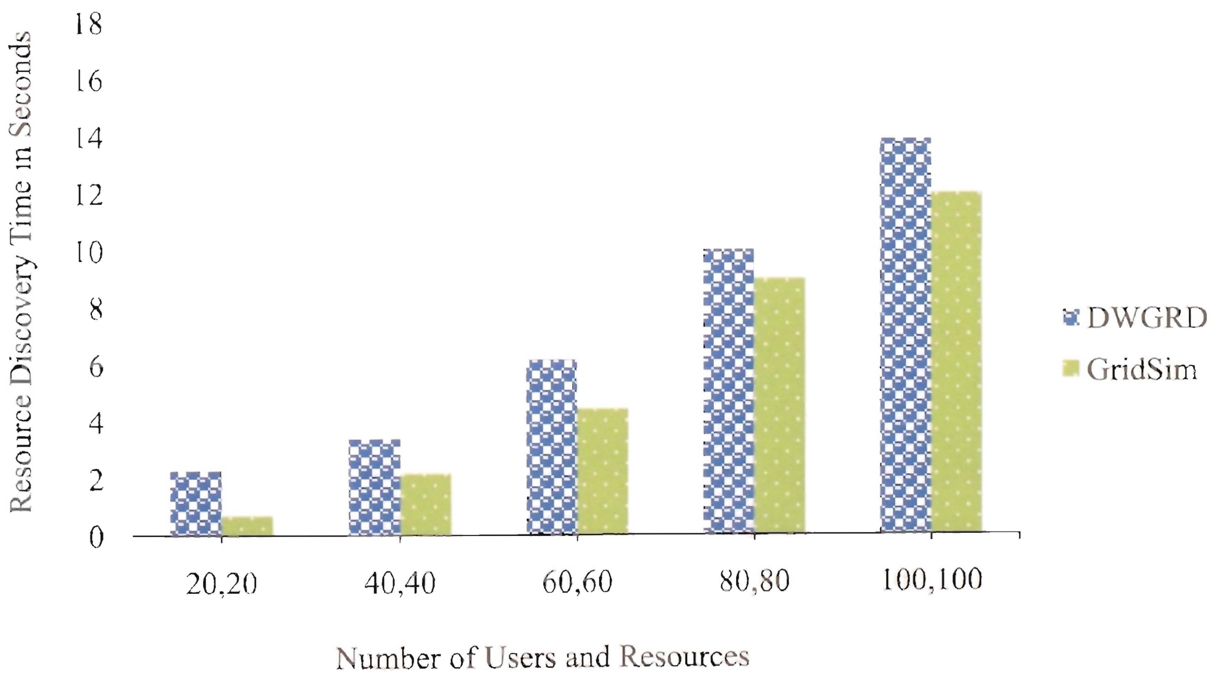


Figure 4.10 Comparison of increase in means of DWGRD and GridSim

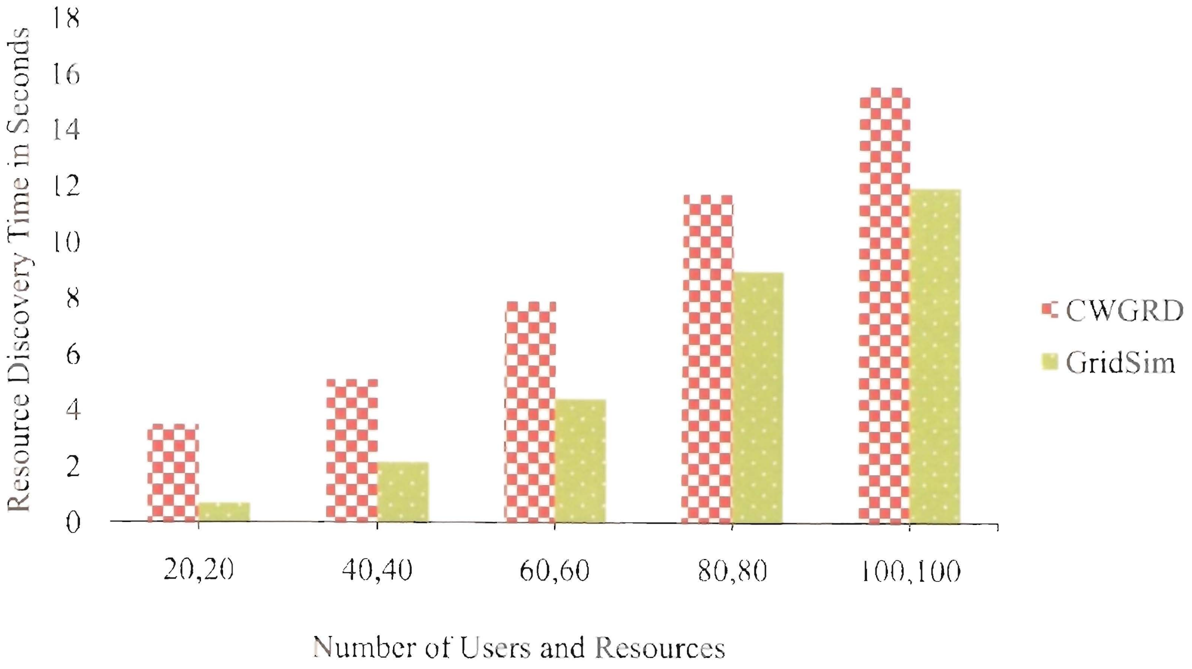


Figure 4.11 Comparison of increase in means of CWGRD and GridSim

After visual comparison we need to do statistical test to see the difference between these means. Considering the low number of the raw data selected for each sample we selected t-test for statistical analysis that is explained below.

4.5. Statistical Analysis

To find the significance of the differences between the means of the resource discovery times of our model (DWGRD and CWGRD) and the resources discovery times achieved by GridSim, we calculated confidence intervals using the t-student distribution and performed t-test for five points shown in the previous graphs. These points show the linear increase in resource discovery time for 20, 40, 60, 80 and 100 users and resources. Please note that users and resources are equal in the number for each point 20, 40, 60, 80 and 100 users and resources. The t-student distribution is used with the following assumptions:

1- Samples of two populations are independent on each other.

2- Samples are drawn from normal populations.

3- The populations have the same variance.

Appendix B shows raw data and the summary of statistics for each point 20, 40, 60, 80 and 100 users and resources in the GridSim, DWGRD and CWGRD models. Since the results of the experiments of CWGRD and DWGRD are independent of GridSim, we did the t-test for unpaired observations and the p-value is recorded that are shown in Tables 4.1 and 4.2. To perform the test, we followed the instructions given by Jain [39] and calculated the required statistics for the 10 observations for GridSim, DWGRD and CWGRD. The statistical results shown in Tables 4.1 and 4.2 are provided by using the equations given below:

$$Mean \quad \overline{x_{a/b}} = \frac{1}{n_{a/b}} \sum_{i=1}^{n_{a/b}} x_{ia/b} \qquad 4.1$$

$$Mean\ Difference \quad \overline{x_a} - \overline{x_b} = \frac{1}{n_a} \sum_{i=1}^{n_a} x_{ia} - \frac{1}{n_b} \sum_{i=1}^{n_b} x_{ib} \qquad 4.2$$

$$Standard\ Deviation \quad S_{a/b} = \left\{ \frac{\left(\sum_{i=1}^{n_{a/b}} x_{ia/b}{}^2\right) - n\overline{x_{a/b}}^2}{n_{a/b} - 1} \right\}^{\frac{1}{2}} \qquad 4.3$$

$$Standard\ Deviation\ of\ the\ Mean\ Difference \qquad S = \sqrt{\frac{S_a{}^2}{n_a} + \frac{S_b{}^2}{n_b}} \qquad 4.4$$

Degree of Freedom for two sample populations$= (n_a + n_b) - 2$ 4.5

$Confidence\ Interval = (\overline{x_a} - \overline{x_b}) \pm t((1\text{-}\alpha/2), (n_a + n_b - 2)*S$ 4.6

Where *a* stands for first raw data population and *b* stands for second raw data population

As it is shown in the tables we used these formulas to calculate mean differences, standard deviation of mean differences for finding t-scores, p-values and the confidence intervals by hand using Excel sheet. The confidence intervals are calculated based on t distribution table given in [40] and presented in Appendix D. The results of our hand calculation are shown on the in Tables 4.1 and 4.2. The similar calculation results obtained from web site offered in [41] are presented in tables C.1 and C.2 of Appendix C.

Table 4.1 **Mean Difference between GridSim and DWGRD Confidence Interval and Unpaired T-test**

Users and Resources	**Mean difference**	**Standard Deviation of Mean difference(s)**	**Confidence Interval 95%**	**p-Value**	**Difference of means.**
20,20	1.573	0.118	(1.326, 1.820)	<0.0001	Different
40,40	1.227	0.478	(0.221, 2.232)	0.0196	Different
60,60	1.744	0.980	(-0.314, 3.802)	0.0918	Not significant
80,80	1.025	1.864	(-2.891, 4.941)	0.5891	Not significant
100,100	1.890	3.721	(-5.927, 9.707)	0.6176	Not significant

Table 4.2 **Mean Difference between GridSim and CWGRD Confidence Interval and Unpaired T-test**

Users and Resources	Mean difference	Standard Deviation of Mean difference(s)	Confidence Interval 95%	p-Value	Difference of means.
20,20	3.289	0.121	(3.036, 3.542)	<0.0001	Different
40,40	2.946	0.475	(1.948, 3.944)	<0.0001	Different
60,60	3.462	0.977	(1.410, 5.513)	0.0023	Different
80,80	2.742	1.865	(-1.176, 6.660)	0.1587	Not significant
100,100	3.606	3.719	(-4.208,11.421)	0.3451	Not significant

Following the Jain [39] we calculated the confidence interval for each comparison. According to these tests, if the calculated interval includes zero or the p-value is greater than 0.05, the difference between two means is not significant.

For calculating p-value we used the instruction given by Mendenhall *et al.* [40] and verified the result with the online statistical service offered in [41]. As shown in the Tables 4.1 and 4.2, 95% confidence interval which is (1-p value) and considering the p-values, we can conclude the following:

1. The difference between the means of resource discovery time of DWGRD and GridSim is significant only when there are equal or fewer than 40 resources and 40 users in the grid.
2. When the number of resources and users are increased to more than 40, the difference between GridSim and DWGRD is not significant.
3. The difference between the means of resource discovery time of CWGRD and GridSim is significant when the numbers of resources and users are 20, 20 and 40, 40 and 60,60. This is an expected result when considering the greater value of the mean

of CWGRD resource discovery time compared to the mean of DWGRD resource discovery time. We expect to see more difference between CWGRD to the GridSim than DWGRD to GridSim.

4. When we increase the numbers of resources and users, the differences between the means are not significant.

Finally, the results of the t-test support the scalability of our proposed model. In the simulation the scalability can be tested when we load the grid with more resources and users. Therefore, we expect that the cost (i.e., the resource discovery time) of adding our model compared to the resource discovery time of grid to be marginal when accessing the resource grids through the Internet.

4.6. CONCLUSIONS

In this chapter, the Direct Web-Based Grid Resource Discovery model including user interface is implemented in NetBeans IDE 6.5.1. We performed the simulation for all three models and did not find any abnormality in the simulation results. All three models show the stability in the results. After that we look at the cost of models as increase on grid resource discovery when the number of resources and users increases in the grid. We found that when increasing the load on the grid, the cost of our model decreases and finally with the grid load of 80 users and 80 resources and more than that, the cost of our model in comparison to grid resource discovery time is marginal. This result supports the scalability of proposed model.

CHAPTER 5

5. CONCLUSIONS AND FUTURE WORK

This chapter concludes our thesis by explaining our contributions in section 5.1. Future work is mentioned in section 5.2.

5.1. CONCLUSIONS

In this work a web service-based layered model that proposed a distributed search for resource discovery in grid computing is proposed. This model provides a scalable solution for information administrative requirements when the grid system expands through the Internet. Distributed search in this model is based on sending DNS queries to the repositories that are distributed in the Internet. Two architectures required for regional grid resource discovery in the proposed model are implemented. We called them Direct and Centralized Web-Based Grid Resource Discovery architectures. The underlying technologies for these architectures are also introduced.

We called these two architectures as the best-case scenarios of the model because they simulate the successful regional search for the URL of resource finder service which is the used in the model. The DWGRD architecture implemented in this work is obviously the best-case scenario of our proposed model. In this implementation the client knows the resource finder's URL from the beginning. CWGRD implementation is the second implemented architecture that is based on contacting central UDDI database registry located in the same region as the client. In this implementation, the client only needs to find the resource finder's URL through a regional UDDI. We consider this case as the second best-case scenario because we assume the URL of regional UDDI is known by the client and UDDI successfully returns the URL of the required resource finder web service to the client.

The DWGRD structure including user interface is implemented in NetBeans IDE 6.5.1 and used in the simulation together with GridSim as explained in Chapter 4. The slightly higher average time for resource discovery observed in the DWGRD model compared to the GridSim when increasing the number of users and resources, is because of the additional time for using web services to send the simulation parameters and initiating the proper classes to start the GridSim for grid simulation used with DWGRD architecture.

The CWGRD architecture is also including client interface, which is according to the proposed model for resource discovery, is implemented with the same technology as DWGRD. The higher average resource discovery times of CWGRD that are achieved by increasing the number of users and resources in the comparison to the GridSim and DWGRD, indicates that this architecture is adding additional time to the GridSim and DWGRD applications. As explained section 4.4 of chapter 4 it is because CWGRD searches the URL of web service from a central regional database and then acts as DWGRD model. The simulation for finding the total grid resource discovery time is conducted in the same environment for both DWGRD and CWGRD models.

Whenever the URL of the resource finder cannot be found locally that means by contacting regional UDDI there is no resource finder that satisfies client's requirements. In this case our model uses distributed search. In the distributed search, our model uses caching in UDDI repositories similar to Domain Name Servers in DNS protocol, which is why searching time in our model after the first time should be equal to the best-case scenarios (i.e., regional search) in the rest of the cases.

By doing simulation and performing t-test for analyzing the achieved data, we found that when increasing the load of the grid, the resource discovery time (i.e., cost) of our model compared to grid resource discovery time decreases; finally, with the grid load of 100 users and 100 resources, there is no significant difference between the resource discovery time of our model in comparison to grid resource discovery time.

Based on this result, we conclude that if our model were implemented on the Internet with thousands of grid resources and users, the cost of our model is negligible which means the proposed model would be scalable. Even if we add the couple of seconds for DNS-like distributed search to include the worst-case scenarios to the cost of our model, still this time will be negligible compared to resource discovery time of grid in the Internet.

In summary the contributions of this work are:

1. Introduction of a simple and scalable web service-based model for accessing grid resources through the Internet.

2. Introduction of the underlying technology for implementation of this model.

3. Implementation of a prototype of the model that can be added on top of a grid computing simulation tool (e.g., GridSim) for estimation of the total resource discovery time when using the proposed architectures.

4. Performing a simulation to examine the scalability of the model.

5.2. FUTURE WORKS

The limitation of this work is that both the DWGRD and CWGRD architectures are implemented by one client and one server on the same machine as the grid simulator. This work in future can be extended to implement the proposed model with multiple clients and multiple server nodes examining a variety of scenarios of finding the resources distributed in a real grid computing environment.

The proposed model is a layered structure for resource discovery in which only the implementation of a resource finder web service and web service-based user interface in the upper layers of Internet are discussed. Doing research on the implementation and simulation of the other parts of the grid computing in the lower layers to enhance our model is the future direction of this work.

REFERENCES

[1] A. Szalay and J. Gray, "The World-Wide Telescope", *Science*, Vol. 293, No. 553, pp. 2037 – 2040, September 2001, http://www.sciencemag.org/cgi/content/abstract/293/5537/2037

[2] G. Coulouris, J. Dollimore and T. Kindberg, "Distributed Systems: Concepts and Design", Fourth Edition, p. 19, pp. 783-824, Addison-Wesley Press, 2005

[3] B. C. Neuman, "Scale in Distributed Systems", Reading in Distributed Computing Systems, pp. 463-489, Los Alamitos, CA: IEEE Computer Society Press, 1994

[4] W3C, SOAP, http://www.w3.org/TR/soap/, last visited March 14, 2010

[5] W3C, XML, http://www.w3.org/XML/, last visited March 14, 2010

[6] The Network Simulator 2, http://www.isi.edu/nsnam/ns/, last visited March 14, 2010

[7] P. Mockapetris, "Domain Names – Concepts and Facilities", Request For Comments RFC 1034, (An Official Document related to Internet Protocol Standards), Nov. 1987, http://www.rfc-editor.org/rfc/rfc1034.txt, last visited March 14, 2010

[8] A. S. Tanenbaum and M. V. Steen, "Distributed Systems: Principles and Paradigms", Second Edition, pp. 9-15, Pearson Prentice Hall, 2007

[9] The Network Simulator 3, http://www.nsnam.org/index.html, last visited March 14, 2010

[10] P. Mockapetris, "Domain Names-Implementation and Specification", Request For Comments RFC 1035, (An Official Document related to Internet Protocol Standards), Nov. 1987, http://www.rfc-editor.org/rfc/rfc1035.txt, last visited March 14, 2010

[11] B. Krishnamurthy and J. Rexford, "Web Protocol and Practice HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement", pp.150-159, Addison Wesley Inc., 2001

[12] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Physiology of the grid: An Open Grid Services Architecture for Distributed Systems Integration", June 22, 2002. http://www.globus.org/alliance/publications/papers.php, last visited March 14, 2010

[13] E. Benson, G. Wasson, and M. Humphrey, "Evaluation of UDDI as a Provider of Resource Discovery Services for OGSA-based Grids", *In Proceedings of 2006 International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, April 25-29, 2006

[14] F. Xhafa, L. Barolli and A. Durresi, "Batch mode scheduling in grid systems", *International Journal of Web and Grid Services*, Vol. 3, No.1, pp. 19–37, 2007

[15] G. Belalem and Y. Slimani, "A hybrid approach to replica management in data grids", *International Journal of Web and Grid Services*, Vol. 3, No. 1, pp.2–18, 2007

[16] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini, "OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies", *International Journal of High Performance Computing Applications*, Vol. 17, No. 4, pp. 403-416, 2003, http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html, last visited March 14, 2010

[17] X. Li, W. Huang, L. Zha, "The architecture and implementation of the Vega Information Grid", *International Journal of Web and Grid Services*, Vol. 3, No. 4, pp.462–479, 2007

[18] B. Schmidt, "A survey of desktop grid applications for e-science", *International Journal of Web and Grid Services*, Vol. 3, No.3, pp.354–368, 2007

[19] F. Lelli, E. Frizziero, M. Gulmini, G. Maron, S. Orlando, A. Petrucci and S. Squizzato, "The many faces of the integration of instruments and the grid", *International Journal of Web and Grid Services*, Vol.3, No.3, pp.239–266, 2007

[20] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International Journal Supercomputer Applications*, Vol. 15, No. 3, pp. 200–222, 2001, http://www.globus.org/alliance/publications/papers.php, last visited March 14, 2010

[21] Y. Yin, H. Cui and X. Chen, "The Grid Resource Discovery Method Based on Hierarchical Model", *Information Technology Journal*, Vol. 6, No. 7, pp.1090-1094, 2007

[22] H. Sun, J. Huai, Y. Liu, R. Buyya, "RCT: A distributed tree for supporting efficient range and multi-attribute queries in grid computing", *Future Generation Computer Systems,* Vol. 24 Issue 7, pp. 631–643, 2008

[23] E. Horowitz, S. Shahni, "Fundamental of Data Structures", pp. 442-456, W H Freeman & Co, 1983

[24] Z. Xiong, Y. Yang, X. Zhang, D. Yu, L. Liu, "Integrated Agent and Semantic P2P Grid Resource Discovery Model", *Proceedings of the eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/ Distributed Computing*, Vol. 2, pp. 216-220, 2007

[25] M. Beheshti Seyed, M. Moshkenani Sadighi, "Development of Grid Resource Discovery Service Based on Semantic Information", *In Proceedings of 10th Communication and Networking Simulation Symposium (CNS' 07)* of SpringSim' 07, Norfolk, Virginia, USA, Vol. 1, pp. 141-148, 2007

[26] I. Foster and C. Kesselman. "Globus: A Toolkit Based Grid Architecture": The Grid Blueprint for a New Computing Infrastructure, pp. 259-278, Morgan Kaufmann, 1999

[27] R. Buyya, S. Chapin, and D. DiNucci, "Architectural Models for Resource Management in the Grid", *the First IEEE/ACM International Workshop on Grid Computing*, pp. 18-35, Dec. 17, 2000, Bangalore, India

[28] F. Cappello, H. Bal, Toward an International "Computer Science Grid", *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, pp. 3-12, May 2007

[29] A. Takefusa, S. Matsuoka, K. Aida, H. Nakada, U. Nagashima, "Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms," *High-Performance Distributed Computing, International Symposium on Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8 '99)*, pp. 97-104, Aug. 1999, http://ninf.apgrid.org/bricks/, last visited March 14, 2010

[30] H. Casanova, A. Legrand and M. Quinson, "SimGrid: a Generic Framework for Large-Scale Distributed Experimentations", *Proceedings of the 10th IEEE International Conference on Computer Modelling and Simulation (UKSIM/EUROSIM'08)*, pp. 126-131, 2008, http://simgrid.gforge.inria.fr/, last visited March 14, 2010

[31] R. Buyya and M. Murshed, GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, The *Journal of Concurrency and Computation: Practice and Experience (CCPE)*, Vol. 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002, http://www.buyya.com/gridsim/, last visited March 14, 2010

[32] C. L. Dumitrescu, I. Foster, "GangSim: A Simulator for Grid Scheduling Studies", *IEEE/CCGrid*, 2005, Cardiff, UK, Vol. 2, pp. 1151 – 1158, http://people.cs.uchicago.edu/~cldumitr/GangSim/, last visited March 14, 2010

[33] O. Kang and S. Kang, "Web-based Dynamic Scheduling Platform for Grid Computing", *IJCSNS International Journal of Computer Science and Network Security*, VOL.6 No.5B, pp 67-75, 2006

[34] M. Giordano, "DNS-Based Discovery System in Service Oriented Programming", *Advances in Grid Computing EGC 2005, European Grid Conference*, the Netherlands, Vol. 3470, pp. 840-850, 2005

[35] S. Bokhari Syed, A. Abhari, and A. Ferworn, "Implementation of Web Based Grid Resources Discovery Architecture", *In Proceedings of 12th Communication and Networking Simulation Symposium (CNS' 09) of SpringSim' 09*, 2009, San Diego, USA, (The revised version was submitted for re-evaluation on November 19, 2009)

[36] Netbeans, Web Services Learning Trail, http://www.netbeans.org/kb/trails/web.html, last visited March 14, 2010

[37] Apache jUDDI, http://www.apache.org/dist/ws/juddi/2_0RC6, last visited March 14, 2010

[38] A. M. Law, W. D. Kelton, "Simulation Modeling and Analysis", Third Edition, pp. 6-7, 496-502, McGraw-Hill, 2000

[39] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", pp. 209-211, Wiley-Interscience, New York, NY, April 1991

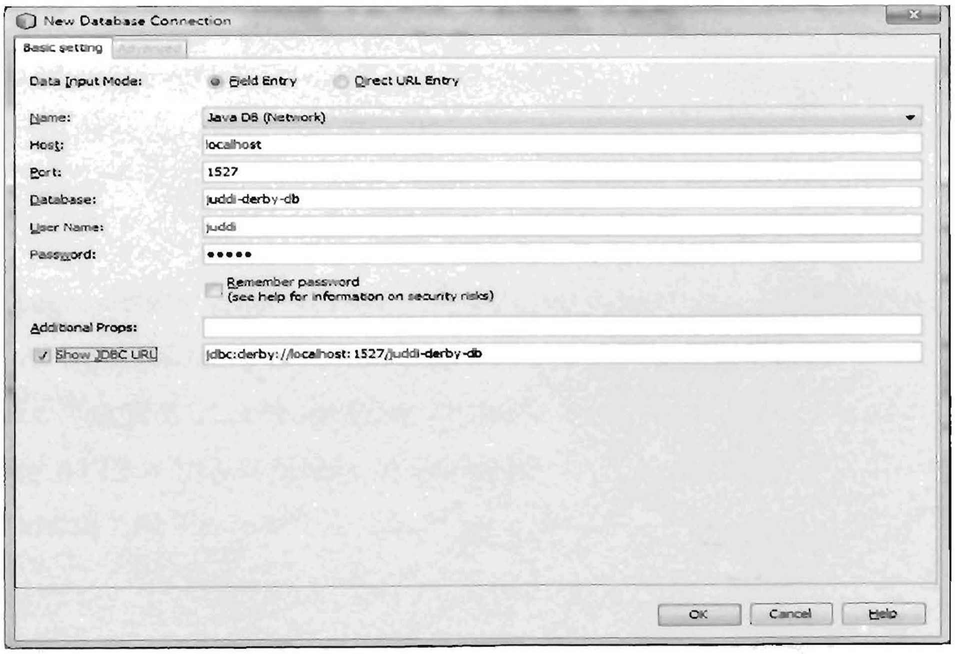
[40] W. Mendenhall, R. J. Beaver, B. M. Beaver, "Introduction to Probability and Statistics", 12th edition, pp.387-410, 691, Thomson Brooks/Cole, 2006

[41] GraphPad Software Inc., http://www.graphpad.com/quickcalcs/ttest1.cfm, last visited March 14, 2010

APPENDIX A

jUDDI Tomcat Server Installation Details

The juddi-tomcat-2.0rc6.zip and juddi-distribution-2.0rc6.zip are downloaded from http://www.apache.org/dist/ws/juddi/2_0RC6/. The juddi-tomcat-2.0rc6.zip presented in [37] is zip file consisting of Apache Tomcat Server 5.5.23 server. The zip file is unzipped into juddi-tomcat-2.0rc6 folder. The juddi version 2.0rc6 is based upon UDDI version 2.0. The folder juddi-tomcat-2.0rc6 contains Apache Tomcat Server 5.5.23 server. The Apache Tomcat Server 5.5.23 server contains different folders such as bin and webapps. The bin folder contains a Derby database folder called juddi-derby-db. The webapps folder contains web applications named juddi and juddi-console. The Apache Tomcat Server 5.5.23 server is installed in NetBeans IDE Version 6.5.1 environment by adding server. The juddi-distribution-2.0rc6.zip file is unzipped and it contains document regarding how to start juddi. The computer is registered with a key as an Apache jUDDI Node with Node Id: uddi:juddi.apache.org: 1467f09b-8652-4a6e-bc45-b08c8b631088 using juddi version 3.0.0.beta.


New Database Connection

Basic setting

Data Input Mode:	[✓ Field Entry] Direct URL Entry
Name:	Java DB (Network)
Host:	localhost
Port:	1527
Database:	juddi-derby-db
User Name:	juddi
Password:	●●●●●
	Remember password (see help for information on security risks)
Additional Props:	
✓ Show JDBC URL	jdbc:derby://localhost: 1527/juddi-derby-db

Figure A.1 Connecting juddi-derby-db database using Java DB (Network) Driver

The derby database juddi-derby-db is copied from bin folder of the Apache Tomcat Server 5.5.23 server which is C:\Users\Saadat Bokhari\Documents\juddi-tomcat-2.0rc6\apache-tomcat-5.5.23\bin into C:\Users\Saadat Bokhari\.netbeans-derby folder. The Databases server Java DB in Java NetBeans IDE Version 6.5.1 is started. The database file juddi-derby-db is connected using Java DB (Network) database Driver as shown in Figure A.1 on the previous page. The database is connected using user juddi and password juddi. The juddi-derby-db database is relational database which consists of 31 tables to keep business, its services, technical detail of business services based upon publisher and his authorizing information. All the information in these tables is linked with each other through some indexing keys.

The Apache Tomcat Server 5.5.23 server is started using service tab of NetBeans IDE Version 6.5.1. The juddi web application already installed in Apache Tomcat Server 5.5.23 server is opened in the browser by right clicking juddi. The browser displays the Welcome jUDDI page (index.jsp) contains three options:

- Validate the local installation's configuration
- jUDDI Console
- Visit the Apache-jUDDI Home Page

The Happy jUDDI page is displayed by clicking Validate link if there is no error in the database configuration otherwise errors are display in the red color. Once the validation is done then it goes back to the Welcome jUDDI page again and by clicking jUDDI Console option goes to juddi-console page.

This page contains a list of three different (Application Programming Interface) APIs such **jUDDI API** (proprietary), **UDDI Inquiry API and UDDI Publish API.** Each API has list of action links. The **jUDDI API** (proprietary) has 4 action links such as getregistryinfo. The **UDDI Inquiry API** has 10 action links to find or get information about business or its services. The **UDDI Publish API** has 16 action links to get authorizing keys or to publish business and its services.

APPENDIX B

Statistical Analysis

B.1 Raw Data and Summary Statistics Obtained Using GridSim when GIS Node=2, Users=20 and Resources=20

Table A **Raw Data in seconds**

1	0.892
2	0.872
3	0.803
4	0.537
5	0.372
6	0.893
7	0.870
8	0.809
9	0.537
10	0.384

Table B **Summary Statistics**

Mean	0.697
Median	0.806
Standard Deviation	0.215
Sample Variance	0.046
Minimum	0.372
Maximum	0.893
Sum	6.969
Count	10

B.2 Raw Data and Summary Statistics Obtained Using GridSim when GIS Node=2, Users=40 and Resources=40

Table A **Raw Data in seconds**

1	2.797
2	2.005
3	3.358
4	0.990
5	1.880
6	0.996
7	2.135
8	2.400
9	2.945
10	2.008

Table B **Summary Statistics**

Mean	2.151
Median	2.072
Standard Deviation	0.772
Sample Variance	0.596
Minimum	0.990
Maximum	3.358
Sum	21.514
Count	10

B.3 Raw Data and Summary Statistics Obtained Using GridSim when GIS Node=2, Users=60 and Resources=60

Table A **Raw Data in seconds**

1	6.540
2	2.569
3	5.064
4	4.042
5	5.296
6	6.141
7	3.076
8	2.593
9	3.841
10	4.992

Table B **Summary Statistics**

Mean	4.415
Median	4.517
Standard Deviation	1.415
Sample Variance	2.002
Minimum	2.569
Maximum	6.540
Sum	44.154
Count	10

B.4 Raw Data and Summary Statistics Obtained Using GridSim when GIS Node=2, Users=80 and Resources=80

Table A **Raw Data in seconds**

1	14.398
2	10.059
3	14.066
4	3.174
5	4.623
6	3.200
7	9.595
8	12.762
9	13.741
10	4.319

Table B **Summary Statistics**

Mean	8.994
Median	9.827
Standard Deviation	4.732
Sample Variance	22.396
Minimum	3.174
Maximum	14.398
Sum	89.937
Count	10

B.5 Raw Data and Summary Statistics Obtained Using GridSim when GIS Node=2, Users=100 and Resources=100

Table A **Raw Data in seconds**

1	25.349
2	19.541
3	4.349
4	3.807
5	7.243
6	3.915
7	7.355
8	18.943
9	4.435
10	25.185

Table B **Summary Statistics**

Mean	12.012
Median	7.299
Standard Deviation	9.126
Sample Variance	83.286
Minimum	3.807
Maximum	25.349
Sum	120.122
Count	10

B.6 Raw Data and Summary Statistics Obtained Using DWGRD Model when GIS Node=2, Users=20 and Resources=20

Table A **Raw Data in seconds**

1	2.720
2	1.958
3	2.149
4	2.428
5	2.433
6	2.578
7	2.536
8	1.868
9	1.971
10	2.054

Table B **Summary Statistics**

Mean	2.270
Median	2.289
Standard Deviation	0.304
Sample Variance	0.092
Minimum	1.868
Maximum	2.720
Sum	22.695
Count	10

B.7 Raw Data and Summary Statistics Obtained Using DWGRD Model when GIS Node=2, Users=40 and Resources=40

Table A **Raw Data in seconds**

1	2.429
2	5.276
3	3.672
4	5.038
5	4.779
6	1.992
7	2.975
8	1.970
9	3.616
10	2.032

Table B **Summary Statistics**

Mean	3.378
Median	3.296
Standard Deviation	1.301
Sample Variance	1.694
Minimum	1.970
Maximum	5.276
Sum	33.779
Count	10

B.8 Raw Data and Summary Statistics Obtained Using DWGRD Model when GIS Node=2, Users=60 and Resources=60

Table A **Raw Data in seconds**

1	2.516
2	4.634
3	6.521
4	8.511
5	8.225
6	4.013
7	9.538
8	3.353
9	4.229
10	10.057

Table B **Summary Statistics**

Mean	6.160
Median	5.578
Standard Deviation	2.756
Sample Variance	7.594
Minimum	2.516
Maximum	10.057
Sum	61.597
Count	10

B.9 Raw Data and Summary Statistics Obtained Using DWGRD Model when GIS Node=2, Users=80 and Resources=80

Table A **Raw Data in seconds**

1	16.304
2	6.722
3	10.579
4	10.868
5	14.735
6	7.089
7	9.118
8	8.028
9	11.428
10	5.317

Table B **Summary Statistics**

Mean	10.019
Median	9.849
Standard Deviation	3.513
Sample Variance	12.345
Minimum	5.317
Maximum	16.304
Sum	100.188
Count	10

B.10 Raw Data and Summary Statistics Obtained Using DWGRD Model when GIS Node=2, Users=100 and Resources=100

Table A **Raw Data in seconds**

1	5.810
2	11.111
3	3.234
4	18.122
5	16.957
6	20.653
7	13.251
8	4.784
9	21.260
10	23.841

Table B **Summary Statistics**

Mean	13.902
Median	15.104
Standard Deviation	7.426
Sample Variance	55.142
Minimum	3.234
Maximum	23.841
Sum	139.023
Count	10

B.11 Raw Data and Summary Statistics Obtained Using CWGRD Model when GIS Node=2, Users=20 and Resources=20

Table A **Raw Data in seconds**

1	4.475
2	3.653
3	3.863
4	4.132
5	4.129
6	4.317
7	4.259
8	3.591
9	3.678
10	3.761

Table B **Summary Statistics**

Mean	3.986
Median	3.996
Standard Deviation	0.315
Sample Variance	0.099
Minimum	3.591
Maximum	4.475
Sum	39.858
Count	10

B.12 Raw Data and Summary Statistics Obtained Using CWGRD Model when GIS Node=2, Users=40 and Resources=40

Table A **Raw Data in seconds**

1	4.184
2	6.986
3	5.381
4	6.746
5	6.472
6	3.732
7	4.703
8	3.704
9	5.325
10	3.742

Table B **Summary Statistics**

Mean	5.098
Median	5.014
Standard Deviation	1.288
Sample Variance	1.660
Minimum	3.704
Maximum	6.986
Sum	50.975
Count	10

B.13 **Raw Data and Summary Statistics Obtained Using CWGRD Model when GIS Node=2, Users=60 and Resources=60**

Table A **Raw Data in seconds**

1	4.274
2	6.328
3	8.232
4	10.216
5	9.920
6	5.753
7	11.263
8	5.080
9	5.939
10	11.764

Table B **Summary Statistics**

Mean	7.877
Median	7.280
Standard Deviation	2.745
Sample Variance	7.535
Minimum	4.274
Maximum	11.764
Sum	78.769
Count	10

B.14 Raw Data and Summary Statistics Obtained Using CWGRD Model when GIS Node=2, Users=80 and Resources=80

Table A **Raw Data in seconds**

1	18.060
2	8.414
3	12.287
4	12.575
5	16.431
6	8.832
7	10.842
8	9.753
9	13.136
10	7.028

Table B **Summary Statistics**

Mean	11.736
Median	11.565
Standard Deviation	3.518
Sample Variance	12.374
Minimum	7.028
Maximum	18.060
Sum	117.358
Count	10

B.15 Raw Data and Summary Statistics Obtained Using CWGRD Model when GIS Node=2, Users=100 and Resources=100

Table A **Raw Data in seconds**

1	7.566
2	12.803
3	4.942
4	19.827
5	18.651
6	22.391
7	14.977
8	6.511
9	22.968
10	25.548

Table B **Summary Statistics**

Mean	15.618
Median	16.814
Standard Deviation	7.420
Sample Variance	55.056
Minimum	4.942
Maximum	25.548
Sum	156.184
Count	10

APPENDIX C

Statistical Analysis Tables

Table C.1 **Mean Difference between GridSim and DWGRD Confidence Interval and Unpaired T-test**

Users and Resources	Mean difference	Standard Deviation of Mean difference(s)	Confidence Interval 95%	p-Value	Difference of means.
20,20	1.573	0.118	(1.32562 , 1.82038)	<0.0001	Different
40,40	1.227	0.478	(0.22194, 2.23206)	0.0195	Different
60,60	1.745	0.980	(-0.31324, 3.80324)	0.0918	Not significant
80,80	1.025	1.864	(-2.89045, 4.94045)	0.5891	Not significant
100,100	1.890	3.721	(-5.92671, 9.70671)	0.6176	Not significant

Table C.2 **Mean Difference between GridSim and CWGRD Confidence Interval and Unpaired T-test**

Users and Resources	Mean difference	Standard Deviation of Mean difference(s)	Confidence Interval 95%	p-Value	Difference of means.
20,20	3.289	0.121	(3.03562, 3.54238)	<0.0001	Different
40,40	2.947	0.475	(1.94935, 3.94465)	<0.0001	Different
60,60	3.462	0.977	(1.41026, 5.51374)	0.0023	Different
80,80	2.742	1.865	(-1.17543, 6.65943)	0.1587	Not significant
100,100	3.606	3.719	(-4.20820,11.42020)	0.3451	Not significant

APPENDIX D

Table D Critical values of t from Biometrika appeared in [41]

d.f.	t.050	t.025	t.010	t.0005
1	6.314	12.706	31.821	63.657
2	2.920	4.303	6.965	9.925
3	2.353	3.182	4.541	5.841
4	2.132	2.776	3.747	4.604
5	2.015	2.571	3.365	4.032
6	1.943	2.447	3.143	3.707
7	1.895	2.365	2.998	3.499
8	1.860	2.306	2.896	3.355
9	1.833	2.262	2.821	3.250
10	1.812	2.228	2.764	3.169
11	1.796	2.201	2.718	3.106
12	1.782	2.179	2.681	3.055
13	1.771	2.160	2.650	3.012
14	1.761	2.145	2.624	2.977
15	1.753	2.131	2.602	2.947
16	1.746	2.120	2.583	2.921
17	1.740	2.110	2.567	2.898
18	1.734	2.101	2.552	2.878
19	1.729	2.093	2.539	2.861
20	1.725	2.086	2.528	2.845
21	1.721	2.080	2.518	2.831
22	1.717	2.074	2.508	2.819
23	1.714	2.069	2.500	2.807

24	1.711	2.064	2.492	2.797
25	1.708	2.060	2.485	2.787
26	1.706	2.056	2.479	2.779
27	1.703	2.052	2.473	2.771
28	1.701	2.048	2.467	2.763
29	1.699	2.045	2.462	2.756
∞	1.645	1.960	2.326	2.576