MultiAgent Based Grid Resource Allocation

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

Master of Engineering
in
Software Engineering

Thapar University, Patiala

By:
Kabir Oberoi
(80631007)

Under the supervision of:
Dr. Seema Bawa

MAY 2008

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Certificate

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

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

(**Kabir Oberoi)**

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

(**SEEMA BAWA)**
Computer Science and Engineering Department
Thapar University
Patiala, INDIA

Countersigned by

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

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

Acknowledgement

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

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

I would also like to thank all the staff members and PhD Scholars Ms Anju Sharma and Ms. Shashi Bhanwar who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

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

Kabir Oberoi

(80631007)

Abstract

Development of the World Wide Web makes it possible for multiple computers to work together in order to solve problems and make the most efficient use of resources. A distributed system is composed of such computers which are separately located and connected with each other through a network. One paradigm for computation using distributed systems is the Grid Computing System. Grid computing is an evolving computing infrastructure a suite of different machines ranging from personal computer to supercomputer is loosely inter-connected to provide a variety of computational capabilities to execute collections of application tasks that have different requirements.

Other paradigm of grid computing is the MultiAgent based Systems in which many autonomous agents interact with the centralized entity to solve the problem. The agents in a MultiAgent system may be distributed on different computers (or nodes), where each computer owns its resources. Although the resources in a multi-agent system are connected by a network, through which number of agents may migrate or leave. A key challenge in multi-agent systems is how to select the best agents among all the agents who can solve the problem with minimum cost and time.

Resource allocation is a problem that arises whenever different resources, which have preferences in participating, have to be distributed among multiple autonomous entities (Agents). MultiAgent Resource Allocation is becoming an area of research in computer science and engineering, and economics. This paper describes the competitive market based economy and a multi-agent based system to find the best place to run a job on the grid. We describe here the auction based protocol for efficient resource allocation in grid environment, the platform that can be used to simulate the environment for multiagent. Various issues related to resource allocation and negotiations have also been addressed in the thesis.

Table of Content

List of Figures

Chapter1.
Introduction

1.1 Distributed Computing

Distributed computing is often used to describe a type of computing in which different computing nodes can simultaneously run an application located on different computers that are connected by a communication network. The main motivation for constructing distributed systems is to obtain large scale resource sharing at low cost. To obtain this goal, we need to interconnect widely distributed heterogeneous resources, which are becoming a necessity for constructing distributed systems. Grid computing has been widely seen as a step beyond the conventional distributed computing, incorporating high-bandwidth, high-speed computing, and large database. It has been seen that resource heterogeneity impacts the resource allocation in many ways, in terms of performance, reliability, robustness, scalability and fault tolerance. So resource allocation strategies for such system should be smart, efficient, robust, scalable, and adaptable.

The distributed computing model is characterized by a collection of *autonomous* processing elements or *independent* computers, interconnected via a network, that are capable of collaborating on a task [1]. The main property of distributed systems is that, it has no regular structure. This means it has no identical processors or identical network interconnecting them. A distributed system can consist of extensively different types of processors connected by different types of network to interconnect them. Another important property of distributed systems is that it has no directly accessible common state (i.e. no shared variables in a shared memory to exchange messages) however it maintains a logical common state to exchange messages between the computation nodes [1]. There are lot many issues, challenges, advantages & disadvantages in distributed computing which are addressed in this chapter.

The objectives that are related to distributed computing are:

- **Resource Sharing**

  Each organization independently maintains local computing and resources independently while sharing some resources over the network. Organization can assemble their resources very effectively using distributed computing by acquiring more processing power from other processors while needed for its computation or donating some resources while they are not being used by them.

- **Scalability**

  Monolithic computing is limited to the capacity of computer in terms of available resources while in contrast distributed computing can provide an easily scalable

- **Fault Tolerance**

  Resources can be replicated to sustain its availability in the presence of failure, means; maintaining backup copies of data on the network so when one fails other copies can be used. However, this term is little vague as it is not possible to build a distributed system that is completely reliable in terms of failure [2] and this topic has received an extensive attention in the research community.

- **Afford ability of computers and availability of network access:**

  Connectivity to the Internet is available and generally affordable, the large number of interconnected computers makes ideal environment for distributed computing. With this, it is possible to get more powerful processing environment which is not possible by a single computer.

1.2 Issues and Challenges in Distributed Computing

In the distributed and dynamic domains, control is distributed among computing nodes without having a centralized control. To achieve maximum efficiency of such large distributed computing system, workloads have to be distributed equally among each node so that some nodes might not fall idle without having jobs to process and other nodes might not be overloaded with jobs to be processed. A distributed system enables distribution of workloads and sharing of resources among the computing nodes in the network. There are lot many challenges that needs to be addressed in order to properly

accomplish the distributed computing executable. The following points will discuss the issues:

1. Allocation and Synchronization Problem:

Problem allocation and resource synchronization are the major problems while implementing distributed system. We here decompose, allocate and describe tasks to different nodes and synthesize partial results among a group of computing nodes.

Handling distributed perceptual information among each node to build efficient co-ordination to maintain decentralized control and finally to obtaining a consistent shared system has always been a focus in distributed system implementation [3].

2. Communication Problem

To collect and deliver messages between pairs of processor is the basic and primary activity of any distributed communication network. This task is performed by using some kind of communication mechanism which works in distributed computing to communicate information between the computing nodes in the network. There are many algorithms available that can be used for communication, but the issue is which one would be more efficient.

3. Load Balancing Problem

Another major issue for distributed computing is load balancing i.e. to distribute workloads equally throughout the network. How to divide roles and tasks to make balance in the system and how to maintain consistency ensuring continuous run in the system is always a great challenge?

Load Balancing is a trivial part of distributed system, lot many issues may arise when the demand for computational power increases. The solution that can be thought for load balancing is to make balance in the system while assigning loads to the systems. Here the main aim of the topic comes to play i.e. using the concept of MultiAgent to solve the problem. The concept is addressed later in other chapters.

1.3 Limitations of Distributed Computing

- **Multiple Failure Point**

Distributed computing many times may tends to failure due to inter-dependency of the processing nodes. Since, all nodes depend on the network for communication, failure of one or more computers might create problem for distributed computing system [4].

- **Security Concerns**

The major challenge for distributed computing is the scale, as the size of distributed computing increases, chances of getting the system vulnerable increases. Due to the decentralized nature of distributed computing, it is more prone to security breaches and illegal access that might affect all the participants of the system [4].

1.4 Grid Computing

Grid computing is coming up as a promising next generation technology, problem solving environment for science, engineering and research area, it has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications [5][6]. Grid computing is an issue resolving environment for utilizing the unused resources and maximizing the resource capability, also it is an innovative approach that leverages existing IT infrastructure to optimize compute resources and manage data and computing workloads.

Grid computing is next step to distributed computing or networking.

What is Grid Computing?

IBM says,

"Grid computing allows you to unite pools of servers, storage systems, and networks into a single large system so you can deliver the power of multiple-systems resources to a single user point for a specific purpose. To a user, data file, or an application, the system appears to be a single enormous virtual computing system [3]."

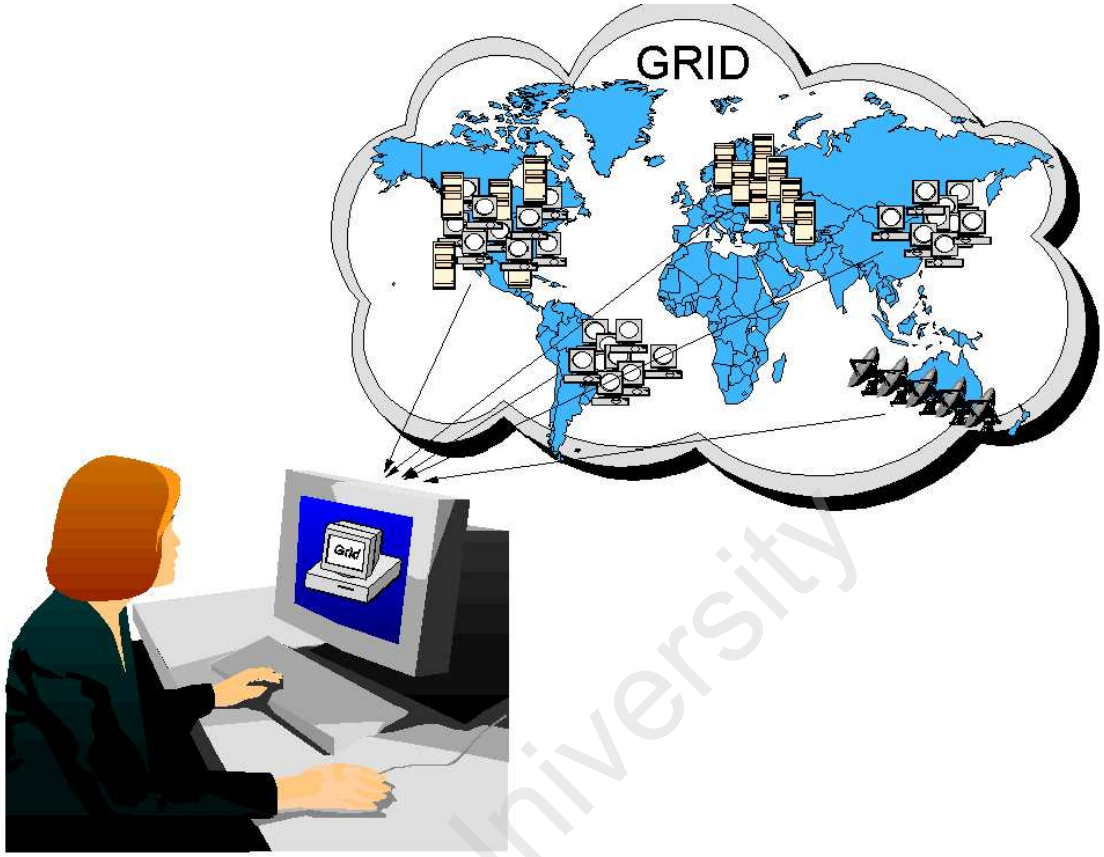


Figure 1.1 The Grid virtualizes heterogeneous and geographically disperse resources for each virtual organization presenting a simpler view [3].

In Figure 2.1 the grid user using the grid, using the distributed computing paradigm to use grid. It has the design goal of solving problems too big for any single supercomputer, at the same time retaining the flexibility to work on multiple smaller problems. Thus grid computing provides a multi-user environment. Multi user environment allows users to utilize the computer and run program at the same time. This implies the use of secure authorization techniques to allow remote users to control computing resources. Grid computing is based on an open set of standards and protocols and enables communication across heterogeneous (computers with different platforms, hardware/software architecture, and computer languages), geographically dispersed environments. It involves sharing heterogeneous resources, located in different places belonging to different administrative domains over a network using open standards. These standards are publicly available specifications for achieving a specific task, by allowing anyone to use the standard to increase compatibility between various hardware and software

components since anyone with the technical know-how and the necessary equipment to implement solutions can build something that works together with those of other vendors.

This large virtual computing system emerges by sharing of resources. This sharing varies from simple file transfer to direct access to computers, software, data and other network accessible resources. The motive of grid is the ability to discover, allocate and negotiate the use of these resources. Grid computing is generally used for problems with requirement of large computational power and/or heavy data storage.

The main characteristics of grid are [7]:

- Multiple administrative domains: Grid spawns into multiple administrative domains. This characteristic makes it different from clusters. Since, it spawns into multiple administrative domains, the policies and autonomy of different domains needs to be maintained.
- Heterogeneity: Grid contains different types of resources like personal computers to super computers to specialized devices like telescopes. Grid provides seamless way to success different resources.
- Scalability: A grid might grow from few integrated resources to millions. The performance might be degraded as the size of grid increases. The application that requires a large number of geographically located resources must be designed to be latency and bandwidth tolerant.
- Adaptability: Grid should be resilient to failure of individual nodes. As it contains a large number of nodes, the probability if failure of nodes will be high. So the applications and the resource brokers behave should adapt dynamically and use the available resources and services efficiently and effectively.

1.5 Motivation

Applying dynamic, heterogeneous and autonomous characteristics of computing resources for grid computing to manage resources while allocation. The heterogeneous nature of resources i.e. resources belonging to different organizations are being used to establish the grid. The jobs are being executed remotely on to the machines, belonging to

different organizations. The scheduling mechanism deploys which jobs are being going to be executed at what time, which machine, etc.

In this thesis, resource allocation problem is addressed, the agent based approach is discussed and MultiAgent based approach is used to solve the resource allocation problem. The agents play an important role in resource selection and resource allocation. The jobs that an agent is going to allocate to resource for execution, both the jobs ad resources in MultiAgent based approach are treated as resources. An agent is a middleman that is going to accept the resource and further submit the job to resource.
The dynamic, heterogeneous and dynamic systems present new challenges in resource management such as: scalability, adaptability and reliability. There are number of questions that arise:

- The comparison between the agent based approach and multiagent based approach, which would be the better approach to follow? Handling many agents or handling a single resource is more economical? Will the multiagent based approach more efficient?
- The systems may get active or go down during runtime without any prior commitment, so the issue is related to this how can we overcome from such problem?
- The adaptability, scalability, reliability issues which are present with grid can they are overcome with the agent based approach?

1.6 Organization of Thesis

This section introduces the organization of thesis:

The First chapter explains the introduction to Distributed Computing. Its advantages & disadvantages, motivation of using distributed computing and its issues. It gives the motivation for the thesis. Simple questions are asked related to the thesis which is addressed in further chapters.

The Second chapter briefly explains the concepts of grid, its relation with the distributed computing. Architecture of the grid, which gives the understanding of the grid, various components of the grid. Introduction to Resource Allocation, various approaches available for resource allocation. Then the terminology MARA, its relation to the Grid and the protocols required to complete the thesis.

The Third chapter covers the problem statement for the thesis. The proposed work for the thesis.

The Fourth Chapter gives the solution to the thesis problem. The pseudo code are given and explained related to the problem statement.

The Fifth Chapter gives the Experimental setup, implementation details, and the Experimental Results in the form of snapshots.

The thesis is been concluded in the Sixth chapter, and the Future work that are related to thesis topic.

Chapter2

Literature Review: Resource Allocation

2.1 Grid Computing

The various terms that are related to Grid Computing are explained:

2.1.1 Components of Grid Computing

There are totally six components of Grid Computing [4]:

1. Security
2. Portal
3. Workload Management
4. Scheduler
5. Data Management
6. Resource Management

2.1.1.1 Security: Grid system and application require the entire standard security functions, including access control, authentication, authorization, integrity, privacy, and nonrepudiation. As grid is growing more and more security issues will arise. Already there are number of security standards being taken by number of middleware's.

2.1.1.2 Portal: When user interacts with the grid environment, it interacts through the application known as Portal, also known as user interface. Here the users are capable of deploying the application that he needs to run. The entire underlying tasks are performed by the portal, i.e. selecting best resource for the job to execute.

2.1.1.3 Application: Applications a user wants to run on a grid must be aware of the resources available. This is where a *workload management* service comes in. An application can communicate with the workload manager to discover the available resources and their status.

2.1.1.4 Scheduler: scheduler is required to locate the resources on which the job needs to be run. Various tasks that a scheduler needs to perform are prioritizing job queues, managing the load, finding workarounds when encountering reserved resources, and monitoring progress.

2.1.1.5 Data Management: In grid environment it's not necessary that data will be available with the system where execution will take place; it might be residing in some other part of world or in the same cluster, so the data management is required to carry data to the right place across the machine, using various protocols.

2.1.1.6 Resource Management: To handle such core tasks as launching jobs with specific resources, monitoring the status of those jobs, and retrieving results, a *resource management* facility is necessary.

2.1.2 Architecture

We need to understand the standards followed by grid, and how grid architecture is defined. There's the architecture OGSA (Open Grid Service Architecture), developed by members of Global Grid Forum (GGF), now known as Open Grid Forum (OGF) [ibm site].

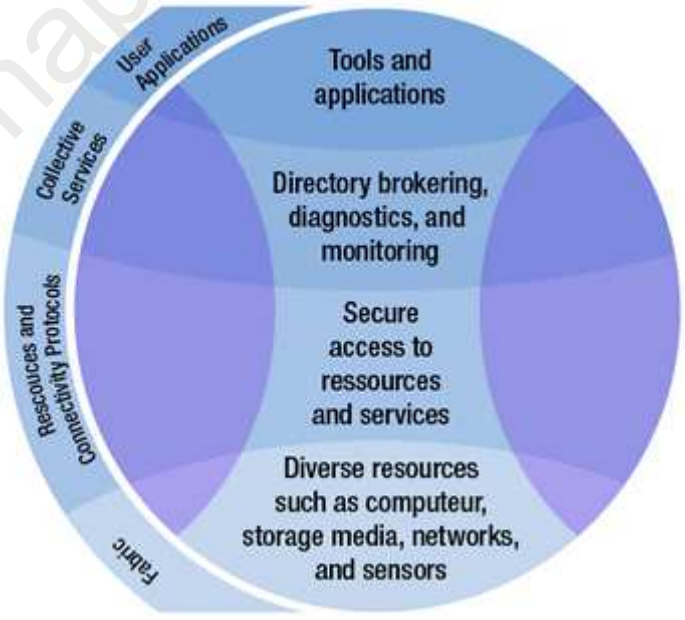


Figure 2.1 Layered Grid Architecture [9]

The architecture defined in Figure 2.2 is the layered architecture, user application at the top which would help the users to use the grid in an easy manner, a number of tools are available to use the grid we call them Portal. Then is the collective services layer, where different middleware comes in role, i.e. brokering, monitoring, management, etc. a connection is required for grid to be operational. So a secure connection is required, to connect to various resources and services. Then finally the fabric layer, fabric layer shows the various resources, media, networks, and the resources that will be contributing to establish the grid.

2.1.3 Types of Grid Systems

The types of grid are mentioned below:

Computational Grid: A computational Grid is a system that aims at achieving higher aggregate computational power than any single constituent machine. A high throughput Grid aims to increase the completion rate of a stream of jobs through utilizing available idle computing cycles [10].

Data Grid: A Data Grid is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data. A Data Grid would allow them to share their data, manage data and manage security issues [10].

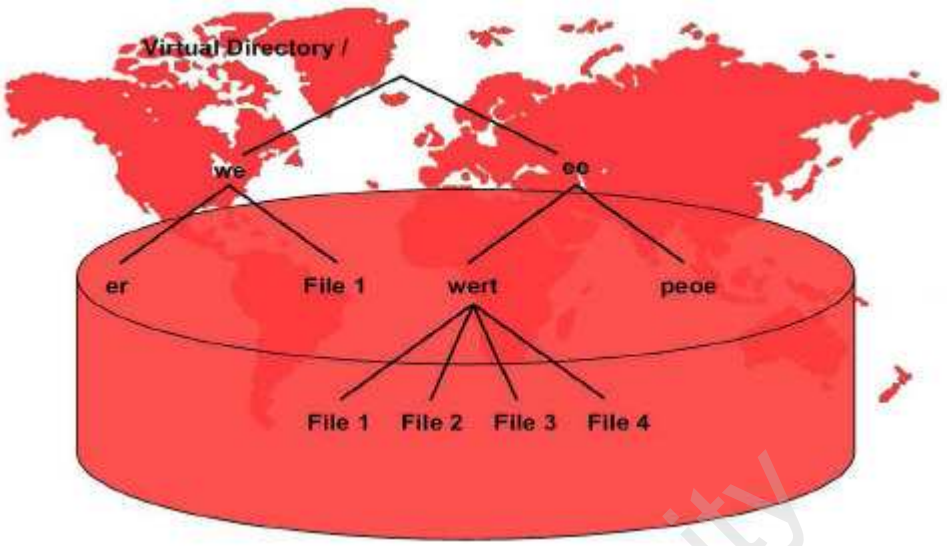


Figure 2.2 Visualization of a Data Grid [11]

2.1.4 Types of Resources

There are a number of resources that are available to form grid, some of them are mentioned below:

Computation: it's the most common and required resource in grid computing as well as general computers. The CPU's can vary in their speed, architecture (X86 or X64), etc. these variation is required in grid computation, since the requirement of the application may vary, and the varying nature of the computation will give rise to grid.

Storage: then is the storage resource, for any of the execution to complete its will require data as input. Storage can be memory attached to the processor or it can be "secondary storage" using hard disk drives or other permanent storage media. Memory attached to a processor usually has very fast access.

Communication: communication speed is increasing every quarter, so the idle bandwidth available can be used for grid. As communication is the base of grid. Better and faster is the communication (intranet or internet), better will be grid. Some different redundant communication path are sometime required, if there is a network failure, or

link breakage. This is must in grid, as there will be chances that many times there will be systems getting active and going down (computers providing resources).

Software's and Licenses: there is much software that is very costlier, which can not be bought. Using the grid, jobs requiring the software are sent to that particular machine, which has software installed. Some software licensing arrangements permit the software to be installed on all of the machines of a grid but may limit the number of installations that can be simultaneously used at any given instant [9].

2.2 Resource Allocation

In the previous topics a lot has been said about grid, where we tried to explain the meaning of grid. In the section 2.1.1 i.e. components of grid we have explained the various terms related to grid. As said resource management is required to handle the jobs, retrieve the results, handling the resources, and many other facilities. It becomes more important in the case where some resources are idle and could be used to relieve overloaded nodes in the same network. Reasons why resource management is necessary in grid are:

First: Available resources should be used maximally or optimally in order to balance the distribution of tasks over the network

Second: If node failure or communication link failure occurs, it is required to re-allocate and manage the tasks and resources from start.

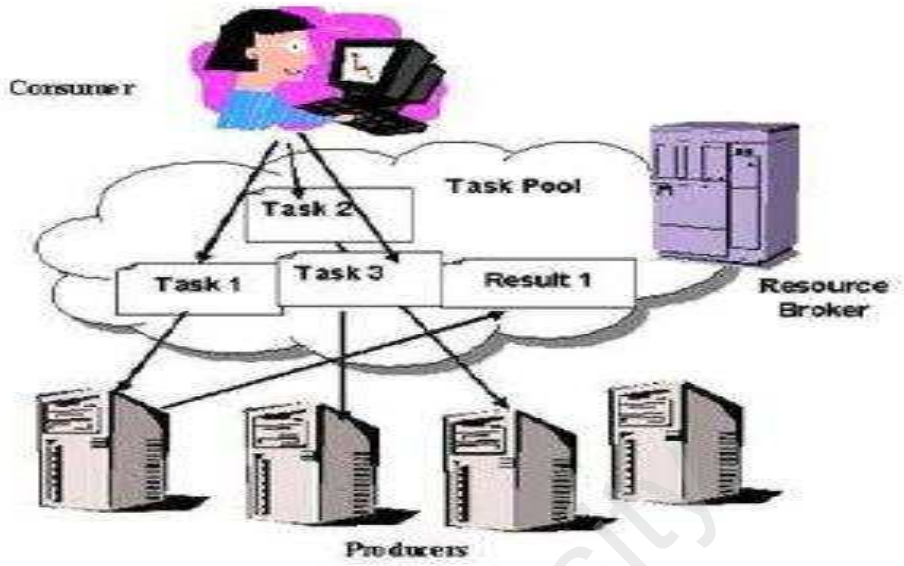


Figure 2.3 Resource Allocation in Grid [12]

Further resource management also needs to take care of resource allocation.

What is Resource Allocation?

"Resource management is the process of managing available resources and system workloads accordingly. The resource allocation is the process of assigning the resources request from jobs to the resources in such a way that it is going to satisfy both the resource capability and jobs [12]".

Resource here includes traditional resources like computing cycles, memory, network bandwidth, space or a storage systems etc which is required by a job for its computation. Resources have their owners who may charge others for using resources. Resources can be shared or exclusive, they can be used for a period of time or may or may not be renewable, refer Figure 2.4.

When the term resource allocation come the jobs needs to be allocated to the resources. So the resources to which the jobs are going to be allocated for execution are

heterogeneous in nature. The resources may be spread world wide, see Figure. 1.1 it describes the heterogeneous nature.

2.2.1 Resource Allocation in Grid

The overall aim of the resource allocation is to utilize the available resources in heterogeneous and dynamic environments. It if often the case in heterogeneous computing network that an application requires multiple resources of different types. It is known that resource heterogeneity impacts the resource allocation in a significant way in terms of performance, reliability, robustness, scalability and fault tolerance. It becomes difficult to implement in a heterogeneous environment where resource requirement and task availability are unpredictable. So the system needs to be smart and adaptable to accept the changes.

To design resource allocation and control mechanism in heterogeneous and dynamic networks the following goals need to be considered [12]:

- Partitioned large complex allocation problems into smaller, disjoint allocation problems
- Decentralized resource access, allocation and control mechanism
- Design reliable, fault tolerant and robust allocation mechanism
- Design scalable architecture for resource access in a complex system
- Provide guarantees to users and applications on performance criteria. Some of the performance criteria in distributed systems include the following:
    - average response time
    - throughput
    - user access time
    - message delay time
    - information loss
    - application failure probability
- Define system performance criteria that reflect in aggregate the diverse individual criteria of users and applications

- Design a unified framework in which users have transparent access to the services of a distributed system, ans services are provided in as efficient manner. This framework should hide the complexity or multiple resource suppliers and resource allocation policies.

Resource allocation and management problem includes:

- Matching problem: the problem of assigning application tasks to suitable resources.
- Scheduling problem: ordering tasks executions in time to optimize a specific objective function.
- Co-allocation problem: can be defined as the problem of simultaneously allocating multiple resources of different types of application in order to meet specific performance requirements.
- Load balancing problem: to increase the utilization of the resources and task in the network.

2.2.2 Resource Allocation Approaches

There are various approaches to allocate resources in a heterogeneous and distributed environment:

2.2.2.1 Matchmaking and Brokering Approach

The matchmaking and brokering approach is the simplest approach. Servers and customers advertise their presence to a common advertising service by describing their characteristics in advertisements. A matchmaker discovers compatible providers and customers with a generic matching operation and notifies the matched agents, which then employ a protocol to connect to each other and enable exchange of service. In this kind of resource allocation approach, there are three different kinds of agent categories involved: service providers, service requester and middle agents [13]. Match making is the process of finding an appropriate provider for the requester through a middle agent. Provider agents advertise their resources to middle agents. Middle agents store these advertisements. A requester asks some middle agent whether it knows of providers with

desired resources. The middle agent matches the request against the stored advertisements and returns the result. While this process at first glance seems very simple, it is complicated by the fact that provides and requesters are heterogeneous and incapable in general of understanding each other.

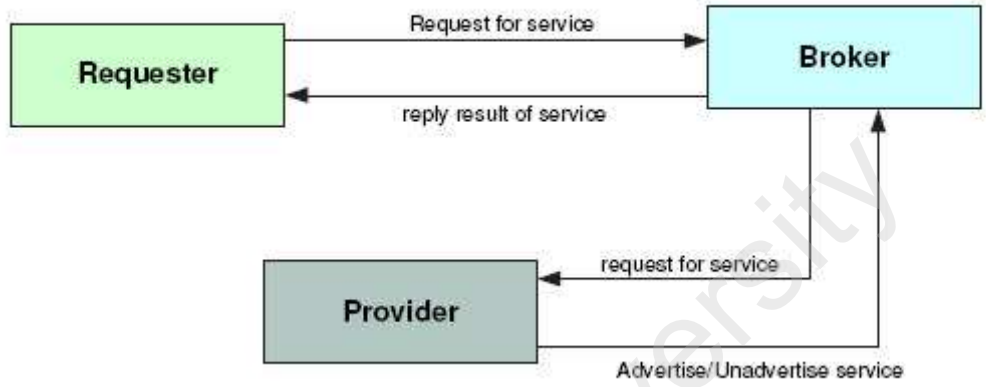


Figure 2.4Matchmaking Approach [13]

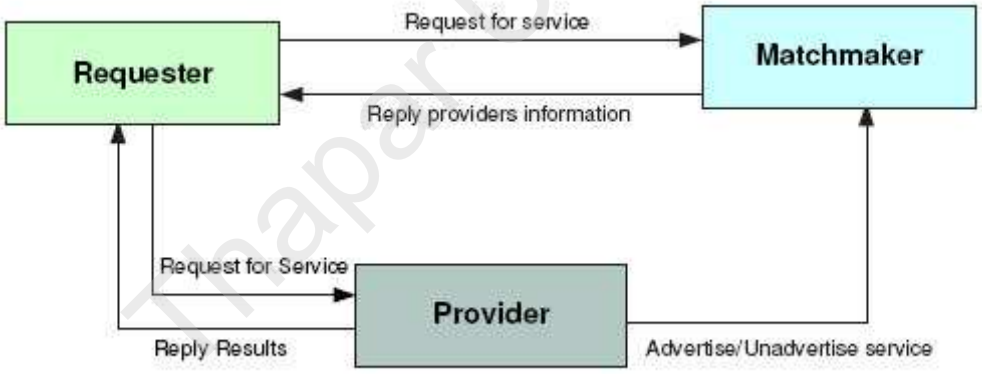


Figure 2.5Brokering Approach [13]

Matchmaking within multiple agents through broker or middle agent is the common and popular approach of resource allocation [14] [15]. One of the major problems facing multiagent systems is finding the services and information that the agent needs and connecting to the agent that is providing this service. There are different types of middle agents such as: matchmaker, broker, billboard etc. In brokering approach, requester request for the service to the broker and broker provides that service on behalf of the

provider. From requester point of view, a broker is indistinguishable from a server. In contrast to broker agent, in matchmaking, a matchmaker doesn't deal with the task of contacting the relevant providers, transmitting the service requester. Figure 2.6 and Figure 2.7 show matchmaking and brokering process.

2.2.2.2 Market Based Approach

In the real world market, there exist various economic models for setting the price of services based on supply-and-demand and their value to users. These real world economy models such as commodity market model, market bidding, auction model, bargaining model etc can also be applied to allocate resources and tasks in distributed computing [16]. In market model resource owners or producers specify their service price and charge users according to the amount of resource consume. Each buyer and seller will set price and quantity of how much it is willing to buy or sell. Consumer pays for their access to various resources they use like CPU cycles, memory, network bandwidth etc. In real world, auctions are used extensively for selling goods/items within a set duration. The most common auction models are one-to-many and many-to-many auction models. There are several auctioning mechanism such as the sealed bid auction, Dutch auction and English auction. In grid economy, decentralization is provided by the fact that economic model consists of agents which selfishly attempt to achieve their goals. Paper [42] provided decentralized algorithms to allocate resource in a co-operative and non-competitive manner among included communication and average processing delay. Resource consumers adopt the strategy of solving their problems at low cost within a required time frame and resource providers adopt the strategy of obtaining best possible return of their investment.

2.3 Multiagent based Resource Allocation (MARA)

The upcoming area in grid Agent based approach for handling the resource allocation. Agents in grid we mean that, an entity that would accept the jobs (tasks or resources) and associate the best possible resource with the job.

MARA based systems emphasizes on resources being distributed among several agents. The preference is given by the agents to the resource that they accept. MARA is the process of distributing a number of items amongst a number of agents [17]. In the following section we define various terms like items, agent, etc., that are related to MARA, which would help us in understanding the MARA concepts in GRID.

- **Resources**

  Here items are the resources that we distribute, and agents receive them. Resources are categorized into divisible resource (different agents may receive different fractions of a resource) and non-divisible resource (it may or may not be possible for different agents to share the same resource).

- **Preferences**

  Agents those receives the resources may or may not have preferences over the bundle they receive.

- **Allocation Procedure**

  The allocation procedure for the resources can be centralized or distributed. In centralized approach a single entity decides the final allocation of resources, and in distributed process the allocation procedure is the result of local negotiation steps. Combinations are also possible: The objective of an allocation procedure is to allocate a resource that is feasible or optimal. The objective may be to find an optimal allocation amongst a small set of feasible allocations; and what is considered optimal could depend both on the preferences of a central entity and on an aggregation of the other agents' individual preferences (e.g. auction mechanisms aiming at revenue maximization and bidding). Of course, computing with an optimal allocation is not possible (due to lack of time; for instance), any progress towards the optimal allocation may be considered a success.

2.3.1 Grid Computing and MARA

One of most recent research topics coming in Grid Computing is MARA [18]. It is true that there are functioning systems for grid resource allocation, but these largely operate in benevolent, cooperative subnets where participants know and trust one another and there is typically no charge for the utilization of resources, although perhaps some artificial accounting system is applied. Such frameworks are exactly what are needed in order to test out many grid middleware functions where the objective is to see a job executed across a range of grid resources. In many respects, grid resource allocation—as distinct from scheduling—and payment for resource usage is a problem to the actual processing of a job.

2.3.1.1 Scalability Issue

If the benevolent, cooperative network of mutually trusting participants is discarded, the client is faced with the problem of piecing together a range of disparate resources that are required to complete the processing of a particular job.

The issues could be seen as a function of scale: Existing grids can handle resource allocation through single centralized mechanisms and (economic) efficiency of allocations may not be important. As grids become larger with a wider range of resources, and used for broader classes of tasks, centralized allocation and inefficient allocation of resources are likely to become less tolerable. In response to this, various approaches need to be evaluated and contrasted under carefully controlled conditions, from centralized systems seeking optimal allocation to distributed mechanisms involving bilateral negotiation. The benchmarks against which to measure the approaches are:

- Centralized systems relying on combinatorial auction clearing algorithms can deliver optimal allocations, but are currently limited by computation costs to hundreds of items and thousands of bids [19].
- Distributed systems relying on bilateral negotiation between consumer and service provider for each component—that is, the consumer constructs their own

bundle—will almost certainly scale, but the results are much less likely to be "good".

2.3.2 Agent based Allocation Procedures

As already discussed in section 1.4, it can be either centralized or distributed. MARA addresses three issues for the allocation procedures [20]:

- Protocols: At this level, we need to address ontological issues (what types of deals are possible?) and devise communication protocols accordingly (what messages do agents save to exchange to agree on one such deal?)[20].
- Strategies: when designing individual agents , we need to devise strategies for agents that allow them to best exploit a given negotiation that they provide incentives to the negotiating agents to adopt a particular desirable profile of behavior (mechanism design) [20].
- Algorithms: At this level, we need to provide algorithms to solve the computational problems faced by agents when engaged in negotiation. This includes both algorithms to decide how to respond to a proposal in a distributed negotiation scenario and winner determination algorithms for combinatorial auctions. Again, this level may provide feedback to the other two levels: If a particular computational problem proves too hard to be solved in a reasonable amount of time then this may call for a simplification of the negotiation protocol (or strategy) [20].

2.3.2.1 Agent Modeling

There are many decisions that need to be taken in order to compute an optimal resource allocation schedule. So we need agents that have a learning system that if any similar types of job come for execution, based on the previous available knowledge of the resource, applies the knowledge to execute the job, and giving the optimal results back to the grid user. Agents would always try to maximize their expected utility [21].

2.3.3 Types of Resources in MARA

The nature of every resource is different, so for resource allocation technique we can consider the nature of resource as the parameter. Every resource vary in their configuration and characteristics. So following are the types of resources discussed:

2.3.3.1 Continuous and Discrete

The resource may be either continuous or discrete. The property of resource will typically influence the trading of resource. For the particular time, the continuous resource will be regarded as infinitely divisible. Individual units of a discrete resource, however, are always indivisible. A continuous resource may be *discretised* by dividing it into a number of smaller parts to be traded as indivisible units. This means that methods developed for discrete MARA are often also applicable in the continuous case.

2.3.3.2 Divisible or Indivisible

As discussed above, resources may treat as being either *divisible* or *indivisible*. While being either continuous or discrete is a property of resources themselves, the distinction between divisible and indivisible resources is made at the level of the allocation mechanism. In this survey, we concentrate on indivisible resources.

2.3.3.3 Sharable or Non-sharable

A *sharable* resource can be allocated to a number of different agents at the same time. While the non-sharable resources at a time cannot be allocated to more then one agents, in complete thesis this assumption is made.

2.3.3.4 Static or Not

A resource may be *consumable* in the sense that the agent holding the resource may use up the resource when performing a particular action. Also, resources may be *perishable*, in the sense that they may vanish or lose their value when held over an extended period of time. We call resources that do not change their properties during negotiation process static resources. In general, resources cannot be assumed to be static. The resource may

be either consumable or perishable but we will consider the case that resource remains static throughout the process.

2.3.4 Protocols

There are various protocols designed, as described below:

2.3.4.1 Auction Protocol

Auctions [6, 7] are centralized mechanisms for the allocation of goods amongst several agents. Agents report their preferences and wait for the final allocation to be made by the auctioneer (whether there is an initial allocation of goods, as in combinatorial exchanges, or not, as in regular combinatorial auctions). The act of reporting preferences is called bidding and, naturally, agents are not required to reveal their true preferences during bidding, but they may submit whatever bid(s) they believe to best serve their own interests. Bidding may be public as in the well-known English auction model or private (sealed bids). In the case of public bidding, we can further distinguish between ascending bids (English auction) and descending bids (Dutch auction). In combinatorial domains, which is what we are interested in here (i.e. there are many goods and agents can submit bids for different combinations of goods), typically, most auction protocols foresee only a single round of bidding using sealed bids. The bidding language determines what types of bids are admissible (and how to interpret them). The auction protocol also specifies which agent would be awarded which goods, based on the bids received in time, and what price should be paid for the bundles allocated to them. In some cases, this decision can be left entirely to the auctioneer (who will seek to maximize her revenue). In other cases, it is important that the auctioneer follows the rules specified by the protocol, as these rules have been designed in such a way as to provide incentives to the bidders to bid truthfully. This is the case for the Vickrey auction model [22], and its extensions to combinatorial scenarios, where the winning agents are paid the prices that are specified in their bids. For an extensive review of different auction models for resource allocation in combinatorial domains we refer to the forthcoming book on Combinatorial Auctions, edited by Cramton, Shoham and Steinberg [23], and the review article on the same topic by Kalagnanam and Parkes [24].

2.3.4.1 Negotiation Protocol [17]

There are two algorithms that are designed for distributed

- **Contract-Net Protocol**

This was the protocol actually designed for task allocation, but it is also applicable for negotiation in MARA. The protocol consists of four phases, involving manager and bidder [9]:

a. Announcement phase: The manager advertises the resource to a number of partner agents (the bidders).
b. Bidding phase: The bidders send their proposals to the manager.
c. Assignment phase: The manager elects the best bid and assigns the resource accordingly.
d. Confirmation phase: The elected bidder confirms its intention to obtain the resource.

This protocol is a basically a 1-to-many protocol, leads to assignment of single task to one single contractor.

- **Concurrent Contract-Net**

There may arise a situation that when contractor is answering a bid for a resource, another bid may arise and might be contractor may loose the bid. So using concurrent-net protocol this may leads to inefficient results. So we propose an extension scheme that there will be pre-bidding and pre-assignment phase. In this the managers and contractors will temporary accept or reject the bids. So the new phases for this are designed as:

a. After a deal has been temporarily accepted, if the manager receives a better offer, this deal can be turned into temporarily rejected offer. It turns out that when many negotiations are conducted simultaneously, by delaying the final acceptance, better deals (from the manager's point of view) may be negotiated.
b. Contractors can modify their offers many times by making temporary offers. If the contractor receives a better new offer from another manager, it can modify its temporary bids before sending a definitive bid.
c. The pre-bidding phase may be quite long. This as the positive effect of reducing the risk of recommitment.

Chapter 3

Problem Statement

As explained in the previous chapter, there are various approaches for resource allocation including matchmaking, market based approach. All of them are having their own drawbacks, like matchmaking focuses on performance analysis, market based approach focuses on economic model. There are protocols discussed related to market based technique, but they all focuses on cost optimization.

The related approaches of resource allocation can also be used with MARA, for solving the particular set of problems. Like brokering is being used with MARA for resource allocation, to select the best possible resource for job execution. Matchmaking can also be used for resource allocation using MARA. This topic of matchmaking of resource allocation using MARA can be the next interesting area in grid.

The proposed approach focuses on cost optimization and performance related issues related to MARA. Certain negotiations associated with the resource allocation are also shown in implementation.

The concept of MARA is explained, it focuses on allocation of resources either in centralized or distributed manner. Further there are agents who would be allocating the resources in centralized or distributed manner. Every agent has their preference for the bundle (gridlets) that will be received. The focus will be on reducing the cost to complete the execution with minimum time.

So the main objectives of the thesis are:

- Designing the agents that have preference to select the resource (gridlets) for execution.
- Designing the Preference for the Agents that will be used during the resource allocation.

- Selecting the best agent for processing the allocation of resource, that is more and more cost effective.
- Designing the allocation procedure that is optimal and feasible.
- Determining weather the allocation matches the cost and the optimization trade-off.

Next chapter explains the solution to the proposed approach.

Chapter 4

Proposed Multiagent Approach for Resource Allocation

This chapter describes the proposed Resource Allocation Algorithm designed using MultiAgent based approach.

In case of competitive multiagent systems, individual agents may not have preference to declare their private information in order to allow the system to select its chosen outcome. Instead, they may be able to lie about their information in a way that will cause the system to select an outcome which they prefer. Secondly, there is no unequivocal notion of optimality in a competitive multiagent system.

There are further lots many problems that can be formulated for resource allocation, given limited resources and many execution entities. Further, we consider in our proposed work that, Agents are the problem solvers for resource allocation. The centralized and distributed based approaches are discussed in the previous chapters. Here we will be using the centralized based approach. A single entity would govern the selection of agents, to whom the jobs (resource) would be submitted for execution. It would be a tedious task to associate the jobs with their appropriate required resources. In grid environment there are two basic issues that need to be considered. The first one is optimal control of each problem-solving agent, and second is preference needs to be considered, while agents accept the gridlets for execution.

4.1 Proposed Work

In Grid layers explained, the resource allocation scheme proposed here lies above the GridSim toolkit layer i.e. resource modeling and simulation layer. The proposed approach has been embedded with the broker, which assigns jobs to various Agents. Resource allocation approach with MARA consists of resource providers, service providers, users and auctioneer for MARA. These agents try to maximize the profits to the organizations.

Resource Allocation starts with submission of jobs. The user submits the job; further best possible matching resource is searched for the job and then the job is submitted to execute. Resource providers provide the resources available with them. Service provider provides the services required executing the jobs; it broadcast services with the help of resource providers. Finally the auctioneer, it tries to reduce the cost of the resource, which will be selected to execute the job. Protocols are implemented to reduce the cost of the resource.

- Users: submits the jobs, which are further send to the broker. User aspects the cost that will be required to completely execute the application.
- Resource Provider: provides the resources, here we have worked. All the Agents are designed, which we call them resource providers. Agents based on the job to execute, formulate the cost for the service and the resources that will be required to execute the job.
- Service Provider: provides the services that are required to completely execute the job. The cost to execute the job using the service is given to resource provider which further adds the cost of the resources and sends back to broker.
- Auctioneer: regulates the market prices for the jobs based on their length, and tries to reduce the cost per second to execute the job. Auctioneer uses Auction Protocol for bidding. Complete process is explained in further sections.

4.1.1 Design of MARA

The design of the application is shown in Figure 4.2. As we have considered the centralized approach for the allocation of resources, we give sample DAG that gives the sample model of the application and sequence diagram for the auction protocol. The value associated with the node describes the cost of the node and the values associated with the links gives the communication cost. The node 1 is the broker part where the jobs are submitted and further nodes at level 2 are the Agents which are the resource providers and the last level are the computational nodes the service providers which would process the jobs. Refer Figure 4.1 for better understanding.

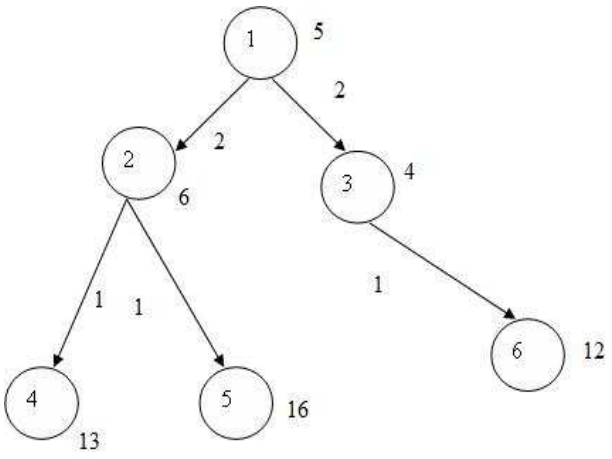


Figure 4.1 DAG model of the Application

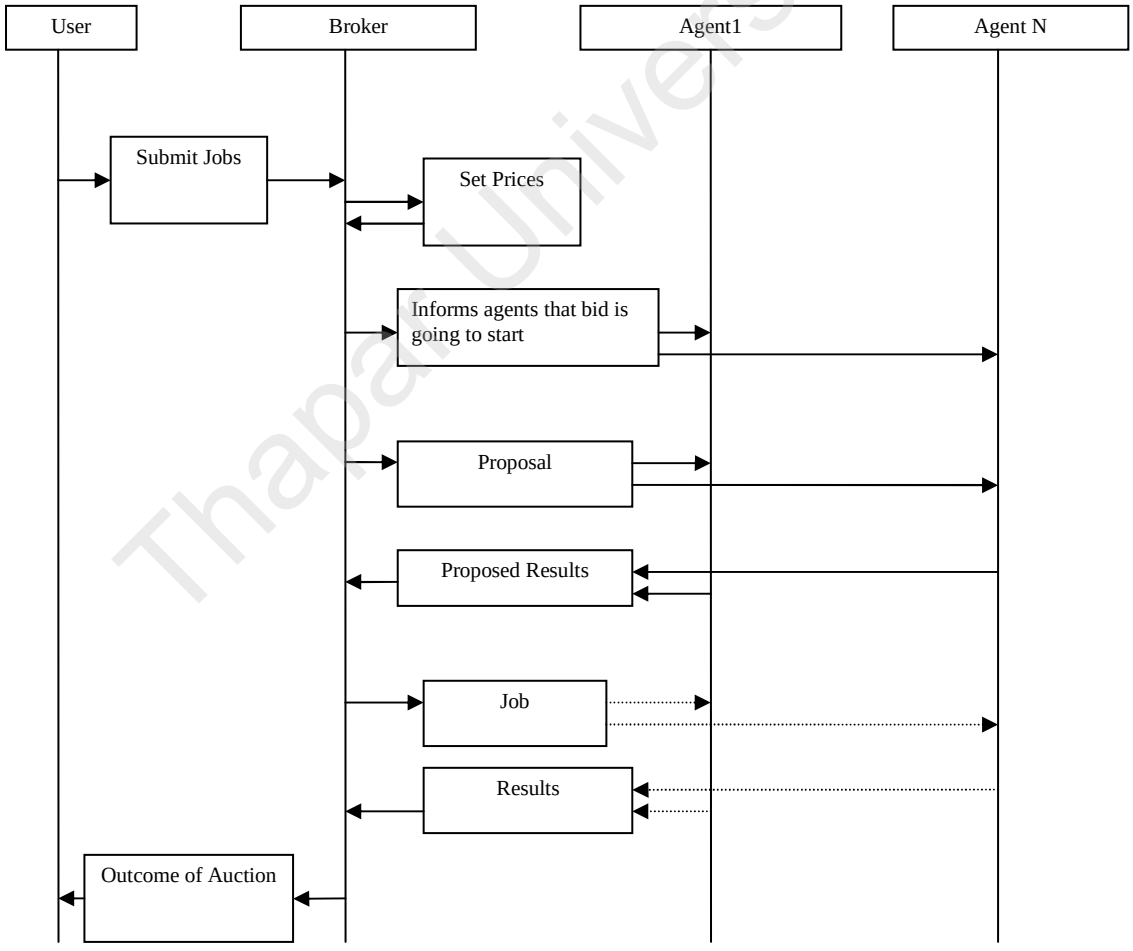


Figure 4.2 Sequence diagram for auction protocol.

The Figure: 4.2 give the sequence diagram for the proposed work, different agents are shown, resource providers, broker the auctioneer and the users. Initially user submits the jobs to the broker. Now the broker is responsible for the submitting and monitoring the jobs on user behalf. The broker creates the auction and sets additional parameters such as job length, auction rounds and the policy that will be used (English or Dutch). As broker also plays the role of auctioneer, the auctioneer informs the agents that the English auction is going to start. Then Auctioneer broadcast the message to for start of bidding. Then each Agent based on the preference further gives the cost for the service and the resource that they will be utilizing for execution. Every agent may or may not participate in bidding; it's based on the preference. Meanwhile, auctioneer keeps on updating the information about the auction. Then number of rounds goes on till a particular agent has been selected. Once the auction clears, Broker informs the outcome to the user.

4.1.2 Proposed Algorithm

Based on the general model of the auctions, we have designed and implemented a generalized auction framework that allows users to develop and evaluate auction protocols for resource management in Grid.

The average Computing cost for the proposed application is, we may find the budget for completing all the jobs.

$$Budget = CMIN + BFACTOR \times (CMAX - CMIN)$$

Where C_{MIN} is the cost of processing all the jobs, in parallel within deadline, giving the cheapest resource the highest priority; *C*MAX is the cost of processing all the jobs, in parallel within deadline, giving the most costly resource the highest priority. An application with $B_{FACTOR} < 0$ would *never* be completed. An application with $B_{FACTOR} \geq 1$ would *always* be completed as long as some resources are available throughout the deadline.

The designed algorithm works in three parts: Job Submission, Agent Selection, and Resource Allocation.

STEP 1: Job Submission

The process starts with job submission; user provides the number of gridlets to be processed. Different jobs with different requirements are created, that have their own requirement to get completed. Now after user lists the number of jobs, it's been submitted to the broker to get it executed. Now it's the Broker's job to get it executed optimally and feasibly.

The pseudo code for the job submission is given below. We have used Queue as the data-structure to store the jobs, which is submitted the Broker.

Pseudo code to create jobs

```
BEGIN
    prompt user for start of application
    read the required number given by user, to create that many number of jobs, N
    if Queue Q is empty
        then
            while N not equal to 0
            do
                create gridlet no. N push it into Queue End(Q)
                decrease N
            end while
    else
        Empty Queue
END
```

Pseudo Code for submission of jobs

```
BEGIN
    prompt user for number of jobs created
    while broker not started
    do
        ping broker continuously
    end while
```

submit complete Queue Q to the broker
while ACK not TRUE
do
monitor for ACK to be TRUE
check weather all gridlets are transferred
end while
END

STEP 2: Agent Selection

Now when the jobs are submitted to the broker, broker has complete list of jobs. Now Broker needs to select the right Agent to get the services and resources. We know that Broker also act as auctioneer, so after getting the gridlets broker creates the configuration and broadcast the message to get appropriate agent. Bidding goes on for particular set of rounds and finally the Agent with lesser cost for providing service and resource is selected. Pseudo code for agent selection is given below:

BEGIN
the first is the assignment phase, in this the broker broadcast the message to all the agents
broker sets the number of rounds n for bidding
while n not equal to zero
do
if n is maximum
broker starts the bidding phase, all the agents based on their preference returns back the bid cost
decrement n
else
agents tries to reduce the cost for the services and resources
agent will lesser cost is identified and selected A
decrement n
end while

broker submits the list of gridlets to the agent A

END

STEP 3: Resource Allocation

Now when agent has been selected for the execution of jobs, now agents needs to submit the gridlets to the resources and services. Agents have services with them which they have used to get the gridlets and will be using to get the results for the job. The scheduling used to allocate the resource is explained in more detail in section 4.2. Pseudo code for resource allocation is given below.

BEGIN

agent having the Q queue of jobs
get the Q length l (number of gridlets)
while all jobs not completed
do
identify the machine in the cluster with free PE (Processing Element)
identify a suitable PE in the machine and assign the gridlet
set the status of the allocated PE busy
decrement l
end while

END

4.2 Simulation of Scheduling

There are simulation environment available as discussed in precious chapters, we have used GridSim as the test bed to test our application, and its explained in more detail in next chapter. There are two types of scheduling algorithm available *simulation in time-shared resource and simulation in space-shared resource.* We have used the space-shared scheduling algorithm to implement the application.

When a job arrives, space-shared systems start its execution immediately if there is a free PE available, otherwise, it is queued. During the Gridlet assignment, job-processing time is determined and the event is scheduled for delivery at the end of the execution time.

Whenever a Gridlet job finishes, an internal event is delivered to signify the completion of the scheduled Gridlet job. The resource simulator then frees the PE allocated to it and checks if there are any other jobs waiting in the queue. If there are jobs waiting in the queue, then it selects a suitable job depending on the policy and assigns it to the PE which is free.

Chapter 5

Implementation Details and Experimental Results

In order to implement our proposed resource allocation model, we developed an Application which uses the simulated the grid environment i.e. GridSim. We have implemented the application using Java and J2EE programming language over the GridSim [25] Toolkit 4.1. The steps followed for implementation is discussed below.

5.1 Technologies Used

In this section we give a brief introduction about the technologies we used in developing our application.

5.1.1 Java

Java language offers several features that ease the development and deployment of a software environment for Grid computing. The first most reason we can give is its open source, and then it's available for most hardware platforms and operating systems. As we need to give the heterogeneity for our application java was the most suited environment due to its byte-code technology.

5.1.2 J2EE

It's the advanced version of java, to design the user interface we have made use of J2EE technology. The applet class, AWT class which helped us a lot in displaying the results to the end user of the application.

5.2 Pre-requisites Required

There are some pre-requisites that are required for the application to function.

5.2.1 Installation of NetBeans IDE [26]

NetBeans is an free, open-source Integrated Development Environment for software developers. You get all the tools you need to create professional desktop, enterprise, web, and mobile applications with the Java language, C/C++, and Ruby. NetBeans IDE is easy

to install and use straight out of the box and runs on many platforms including Windows, Linux, Mac OS X and Solaris.

The NetBeans IDE 6.1 release provides several new features and enhancements, such as rich JavaScript editing features, support for using the Spring web framework, and tighter MySQL integration. This release also provides improved performance, especially faster startup (up to 40%), lower memory consumption and improved responsiveness while working with large projects.

Before installing NetBeans 6.1 make sure that Java 1.4 or later is installed on the system. The NetBeans setup is freely available, which can be downloaded and installed on the system.

5.2.2 Installation of GridSim Toolkit

GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. The GridSim toolkit supports modeling and simulation of a wide range of heterogeneous resources, such as single or multiprocessors, shared and distributed memory machines such as PCs, workstations, SMPs, and clusters with different capabilities and configurations. It can be used for modeling and simulation of application scheduling on various classes of parallel and distributed computing systems such as clusters [27], Grids [4], and P2P networks [28]. The GridSim toolkit provides facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations, and domains. It supports primitives for application composition, information services for resource discovery, and interfaces for assigning application tasks to resources and managing their execution. These features can be used to simulate resource brokers or Grid schedulers for evaluating performance of scheduling algorithms or heuristics. We have used the GridSim toolkit to create a resource broker that simulates Nimrod-G for design and evaluation of deadline and budget constrained scheduling algorithms with cost and time optimizations.

5.2.2.1 Features of GridSim [29]

- It allows modeling of heterogeneous types of resources.
- Resources can be modeled operating under space-or time-shared mode.
- Resource capability can be specified in the form of MIPS (Million Instructions per Second) or as per SPEC (Standard Performance Evaluation Corporation).
- Resources can be located in any time zone.
- Weekends and holidays can be mapped depending on resource's local time to model non-Grid (local) workload.
- Resources can be booked for advance reservation.
- Applications with different parallel application models can be simulated.
- Application tasks can be heterogeneous and they can be CPU or I/O intensive.
- No limit on the number of application jobs that can be submitted to a resource.
- Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be time-shared or space-shared. Network speed between resources can be specified.
- It supports simulation of both static and dynamic schedulers.
- Statistics of all or selected operations can be recorded and they can be analyzed using GridSim statistics analysis methods.

5.2.2.2 System Architecture

We employed a layered and modular architecture for Grid simulation to leverage existing technologies and manage them as separate components. A multi-layer architecture and abstraction for the development of GridSim platform and its applications is shown in Figure 5.1.

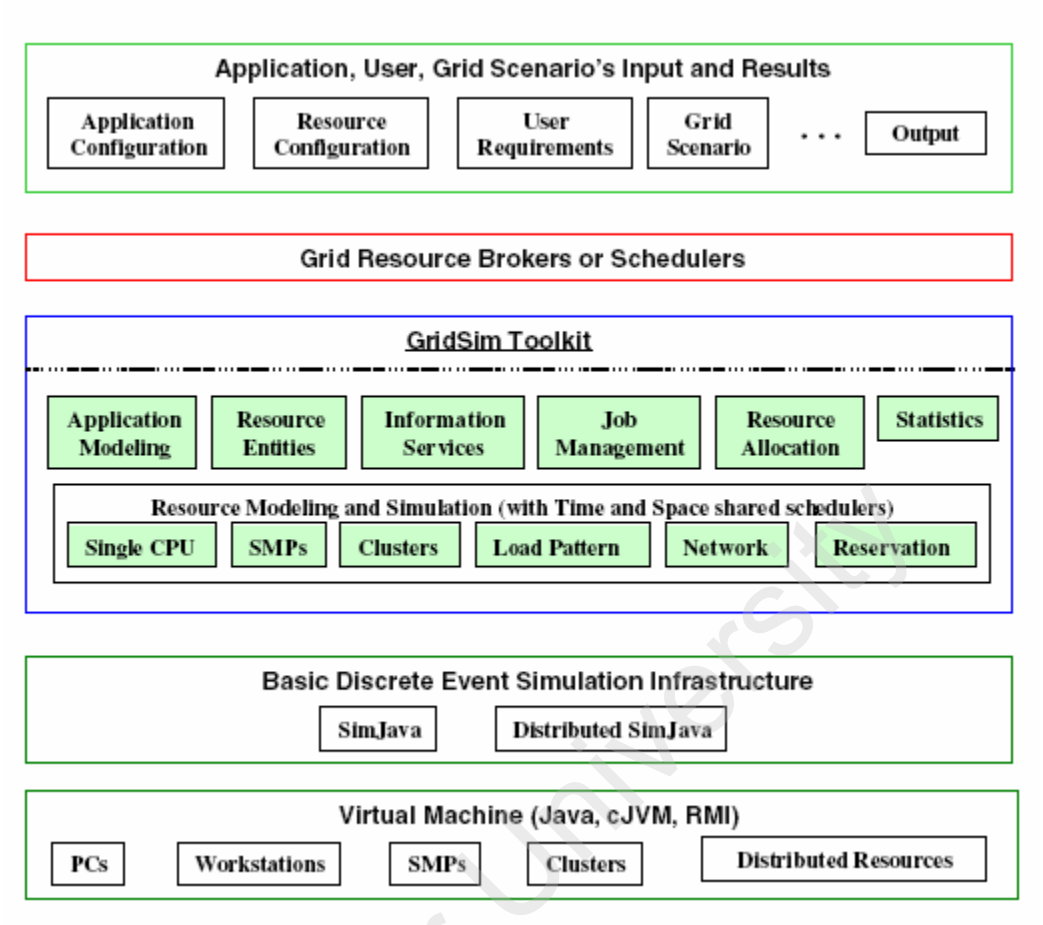


Figure 5.1 Architecture for GridSim platform and components [29]

The first layer is concerned with the scalable Java interface and the runtime machinery, called JVM (Java Virtual Machine), whose implementation is available for single and multiprocessor systems including clusters [30].

The second layer is concerned with a basic discrete-event infrastructure built using the interfaces provided by the first layer. One of the popular discrete-event infrastructure implementations available in Java is SimJava [31].

The third layer is concerned with modeling and simulation of core Grid entities such as resources, information services, and so on; application model, uniform access interface, and primitives application modeling and framework for creating higher level entities. GridSim focuses on this layer that simulated the environment in an heterogeneous way.

The fourth layer is concerned with the simulation of resource aggregators called Grid resource brokers or schedulers.

The final layer is focused on application and resource modeling with different scenarios using the services provided by the two lower-level layers for evaluating scheduling and resource management policies, heuristics, and algorithms

Now as we have learned about the GridSim, we are ready to start working with it. GridSim it's an open source platform and freely available at gridbus.org. Steps to install the GridSim package are as follows:

Step1: Unzip the downloaded GridSim package on a particular location on the hard drive. This location will further be used to set the environment variable for the GridSim.

Step 2: Now we need to set the environment variable for the GridSim package. Set path variable for gridsim.jar and simjava2.jar, suppose for example we have extracted the GridSim package in c:\ gridsimtoolkit-4.1.

Step 3: Add value c:\ gridsimtoolkit-4.1\jars\gridsim.jar

Step 4: Add value c:\ gridsimtoolkit-4.1\jars\simjava2.jars

Just following the above steps will make the GridSim working.
Now if we want to execute the application these are the following steps that we need to follow. Windows is been used as the working environment, so the steps to execute the application on DOS prompt are as follows:

To compile the application:
javac -classpath %GRIDSIM%\jars\gridsim.jar;. myapplication.java
To run the application:
java -classpath %GRIDSIM%\jars\gridsim.jar;. myapplication > file.txt

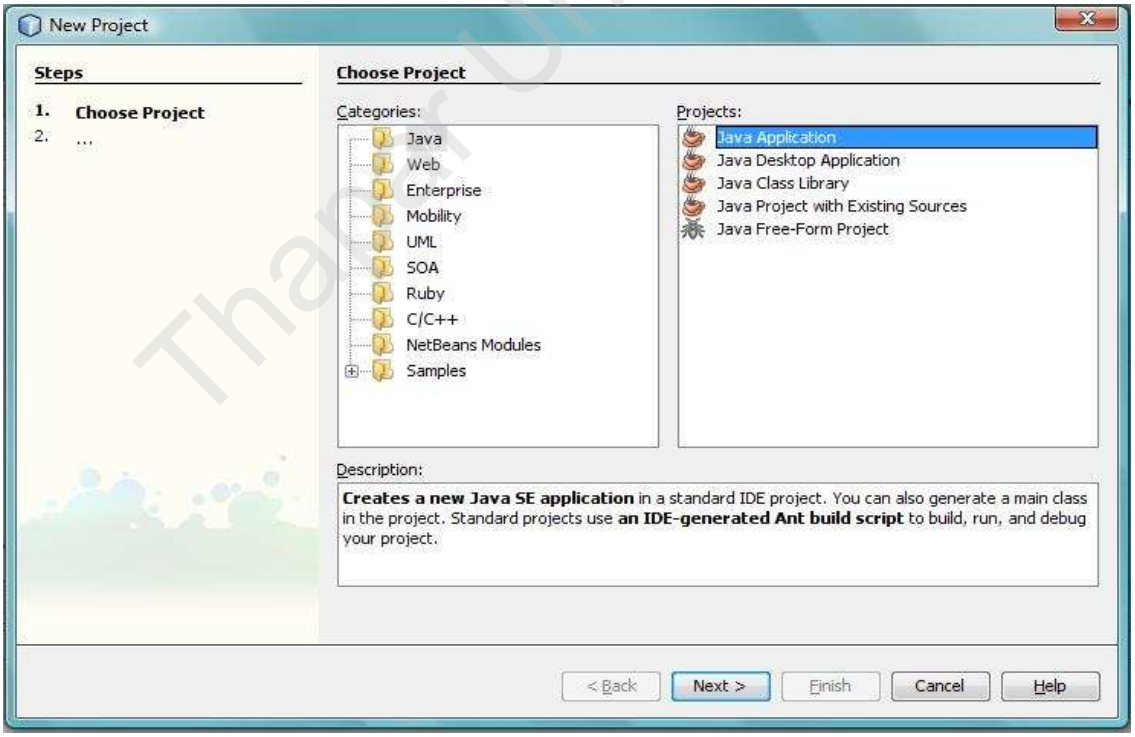
5.2.2.3 Importing GridSim package in NetBeans

We have used NetBeans [26] IDE as the tool to design the application. In previous section we have explained how to compile the application on DOS prompt. To simulate the GridSim environment using NetBeans, we need to import the GridSim package. The application is further simulated over GridSim. The steps are as follows:

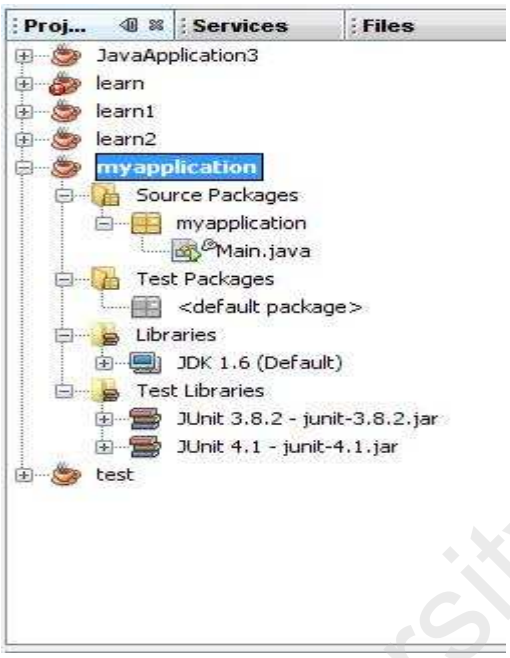
Step 1: Create the new Java Project. Refer Figure 5.2

Step 2: The set of folders along with the package is automatically created. Refer Figure 5.3.

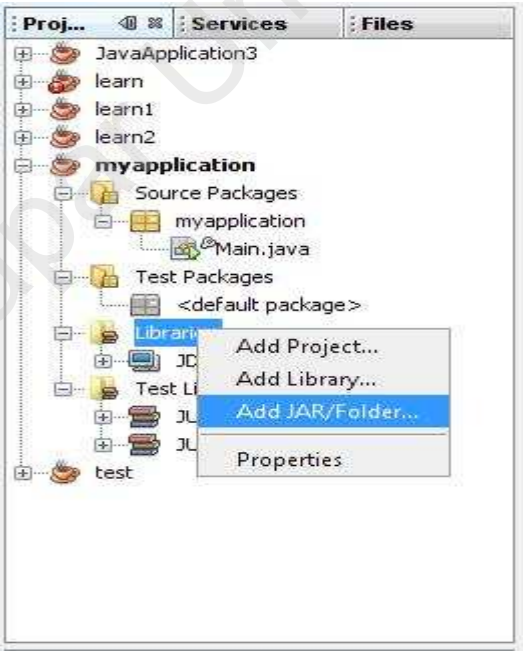
Step 3: In the left side of the window, we have got a tabbed window in NetBeans. There the name of our application exists. There is a hierarchy of folders, we also have a Libraries folder refer Figure 5.3. Right Click the Libraries folder and select "Add Jars/Project". An open window box opens, browse to the location where gridsim package was extracted, and from jars folder select the gridsim.jar file refer Figure 5.4.



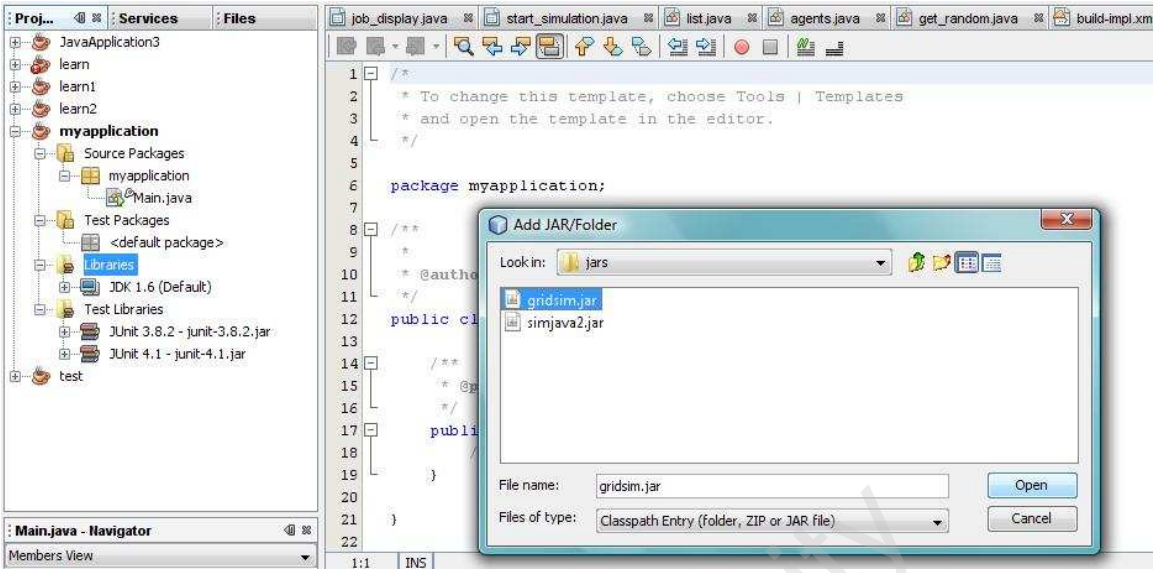
Snapshot 5.1 New Project Window



Snapshot 5.2 Tabbed window showing the Hierarchy of folder associated with our implementation



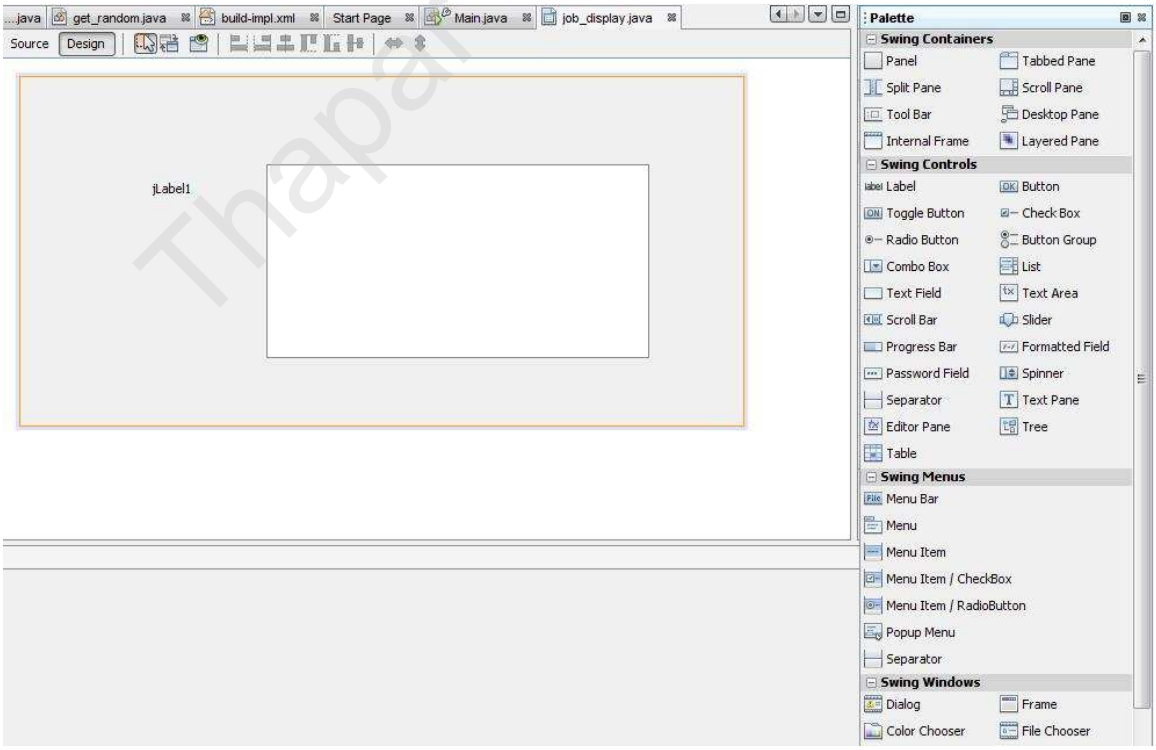
Snapshot 5.3 Add jar Window



Snapshot 5.4 location of GridSim jar package

5.3 Implementation

Till now we have installed all the required components to run the application above the GridSim package. We have used the NetBeans Drag and Drop facility to create the user interface; figure 5.6 shows the NetBeans window.



Snapshot 5.5 Drag and Drop in NetBeans

To completely run the application on GridSim package, we need to follow the steps to create each and every entity required to run the application:

5.3.1 Creating User(s) in GridSim

The users are created in GridSim in a simple manner, there can be more then one user using the grid at a particular time. In GridSim for an application to have more then one user, which needs to be considered differently ID is provided. User ID is going to differentiate between the users.

Simply we need to create ResourceUserList.

ResourceUserList userList = new ResourceUserList();

The above class will create a user list, that will contain the list if users who are active. We need to give the user id for each of the user, with the following command.

userList.add(0)

The figure 5.7 shows the result of creating 2 users.

Snapshot 5.6 Grid Users creation.

5.3.2 Creating Gridlet(s) in GridSim

The job which can run independently and sequentially on a grid resource is called as **gridlet.** Every gridlet has a unique ID, which makes it different from other gridlets. The user provides the number of gridlets that needs to be created, then for each gridlet its length, file size, output file size needs to be specified. Gridlets can be created in a manual way or can be randomly generated. The steps to create gridlets are as follows:

Manual creation:

- Specify the file size, output file size, and length of the gridlet required

'Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);

- Then we have used Queue to keep the list of gridlets, we used link list to keep the jobs together.

GridletList list = new GridleList();

list.add(gridlet1);

- Like this the number of gridlets can be created manually

Randomly generation:

- An object of type random is required
- An object of GridletList is required.

GridletList list = new GridleList();

- GridSimStandardPE needs to be set.
- Randomly we need to generate the length of the jobs, by using the standard PE to MIPS.
- Minimum and Maximum range of the file, that we need to set.
- Following are the values that are created by GridSim

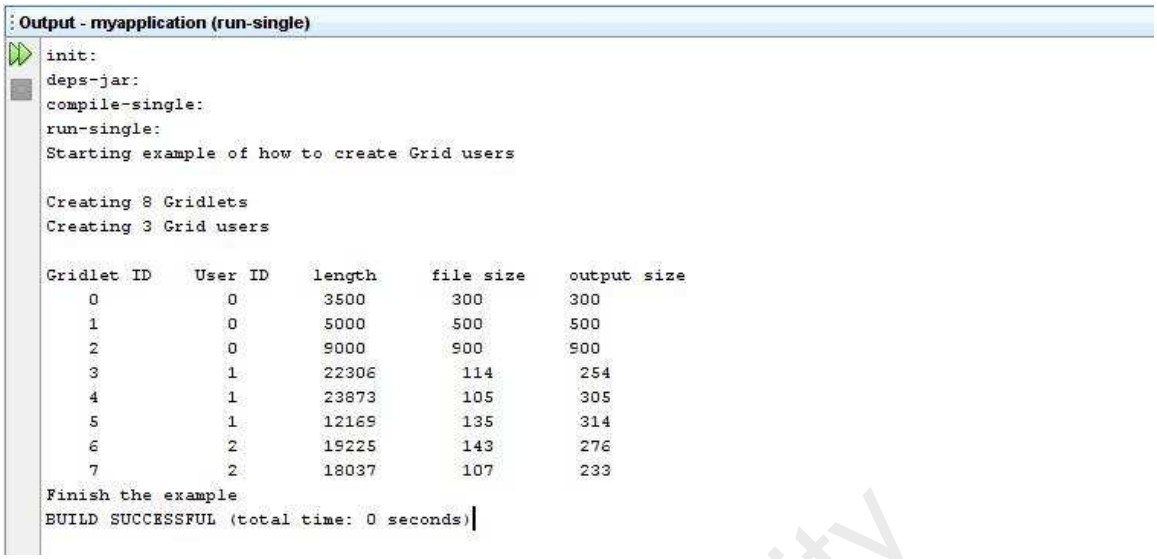
*length = GridSimStandardPE.toMIs(random.nextDouble()*output_size);*

file_size = (long) GridSimRandom.real(100, min_range, max_range, random.nextDouble());

output_size = (long) GridSimRandom.real(250, min_range, max_range, random.nextDouble());

Gridlet gridlet = new Gridlet(id, length, file_size, output_size);

- These lines can be executed as many times we need the randomly generated gridlets.



Snapshot 5.7 Gridlet Creation

5.3.3 Resource Creation in GridSim

The simulated grid environment will be having more then one machine, and every machine can have more then PE (processing Element). The steps required to create grid resource are given below.

- Create the machine list object for the class MachineList.

  MachineList mList = new MachineList();

- Now create PE for the machine, there can be more then one PE.

  PEList peList1 = new PEList();

- Set the MIPS rating for the PE

  MIPSRating = 377;

- Add the processing element to the machine list.

  peList1.add(new PE(1, MIPSRating));

- Create Machine with different ID.

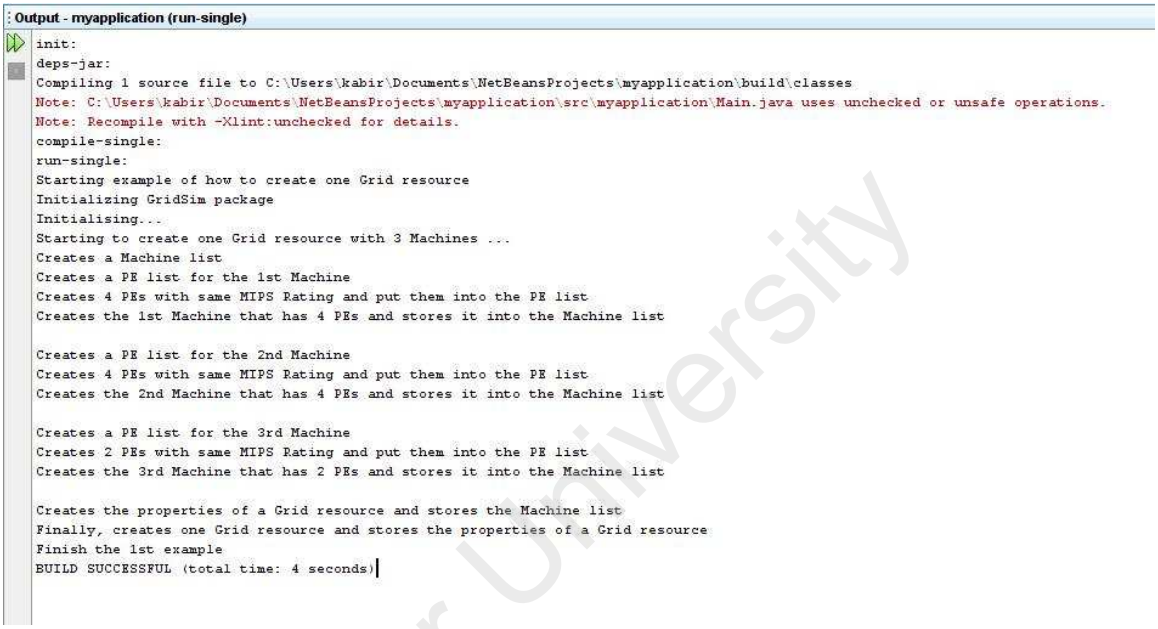
  mList.add(new Machine(1, peList1));

- Repeat all above steps to create the machines.
- Now we need to configure the resources. Different resource characteristics needs to be set, like OS, Architecture, list of machines.

ResourceCharacteristics resConfig = new ResourceCharacteristics(
arch, os, mList, ResourceCharacteristics.TIME_SHARED, time_zone, cost);

- In the final step we need to specify the resource name, communication speed, peak load, off-peak load, holiday list and load along with resource characteristics.

GridResource gridRes= new GridResource(name, baud_rate, seed, resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,Holidays);



Snapshot 5.8 Resource Creation

5.3.4 GridSim Simulation

Now we have all the entities available with us that will make the simulation complete. The following are the steps need to be followed:

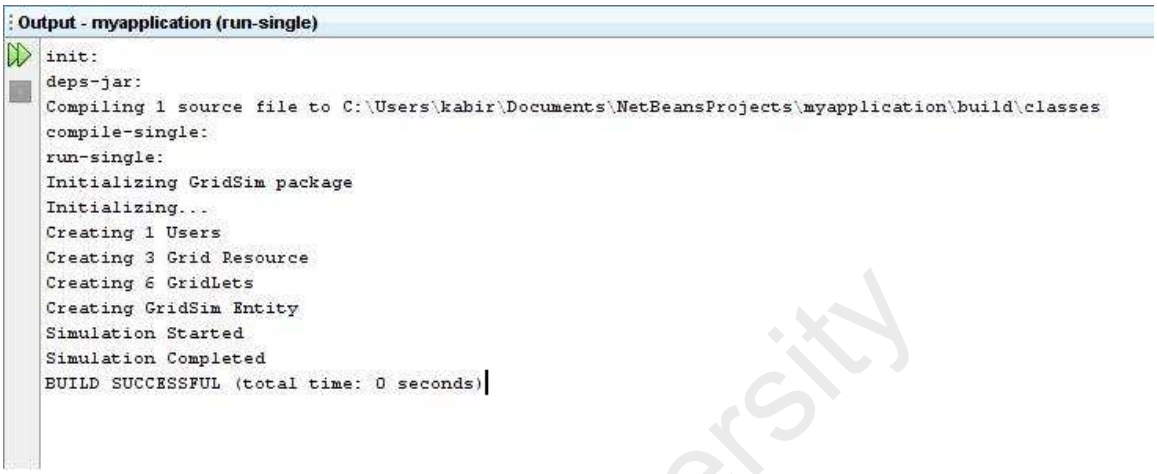
- Calendar Type Instance.
- Trace Boolean Flag with which we can control the tracing of GridSim Events.
- List of files or processing names to be excluded from any statistical measures and report name.

String[] exclude_from_file = { "" };

String[] exclude_from_processing = { "" };

String report_name = null;

GridSim.init(num_user, calendar, trace_flag, exclude_from_file, exclude_from_processing, report_name);

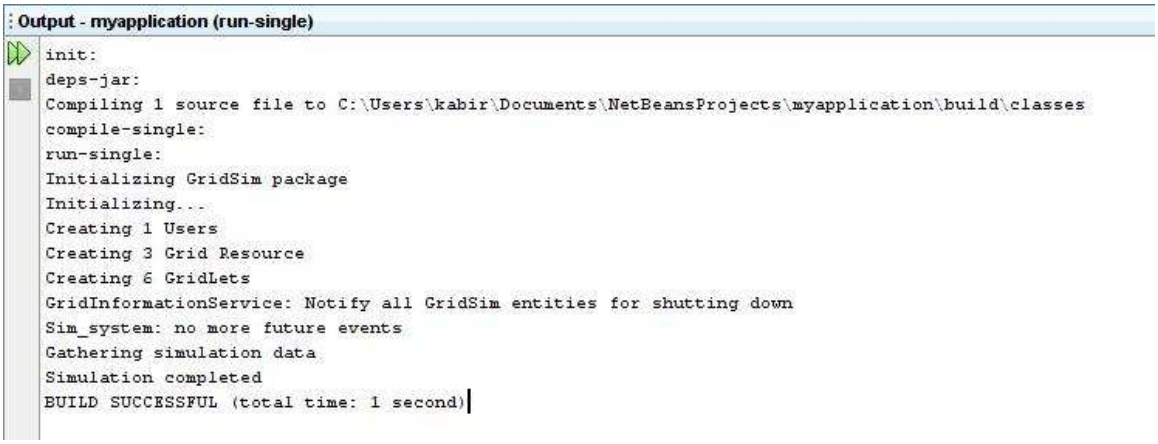


Snapshot 5.9 GridSim Simulation

5.3.5 Shutting Down GridSim

After the simulation is completed, we need to stop the simulation, they internally clear all the buffers, deletes files, releases reports.

- *GridSim.shutdownGridStatisticsEntity();*
- *GridSim.shutdownUserEntity();*
- *GridSim.terminateIOEntities();*



Snapshot 5.10 Stopping GridSim

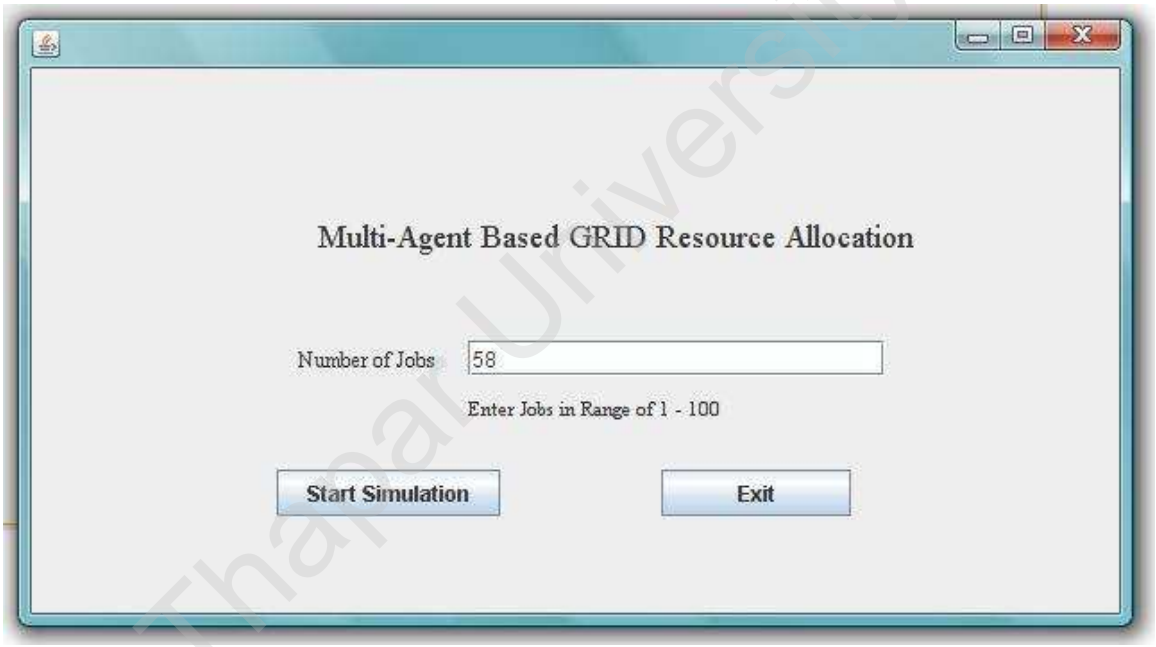
Thus till now we have seen how to create grid resource, grid users, gridlets, and how to start the simulation. Using the knowledge of all the above GridSim utilities, we have developed our application. Next section discusses the experimental results.

5.4 Experimental Results

We have learned how to use GridSim package, to simulate the complete grid environment. In previous section we have shown how to create different entities, that all would collaborate to form a complete application to run. GridSim Version 4.1, Java 1.6, NetBeans IDE 6.1 to completely run the application.

5.4.1 User Inputs

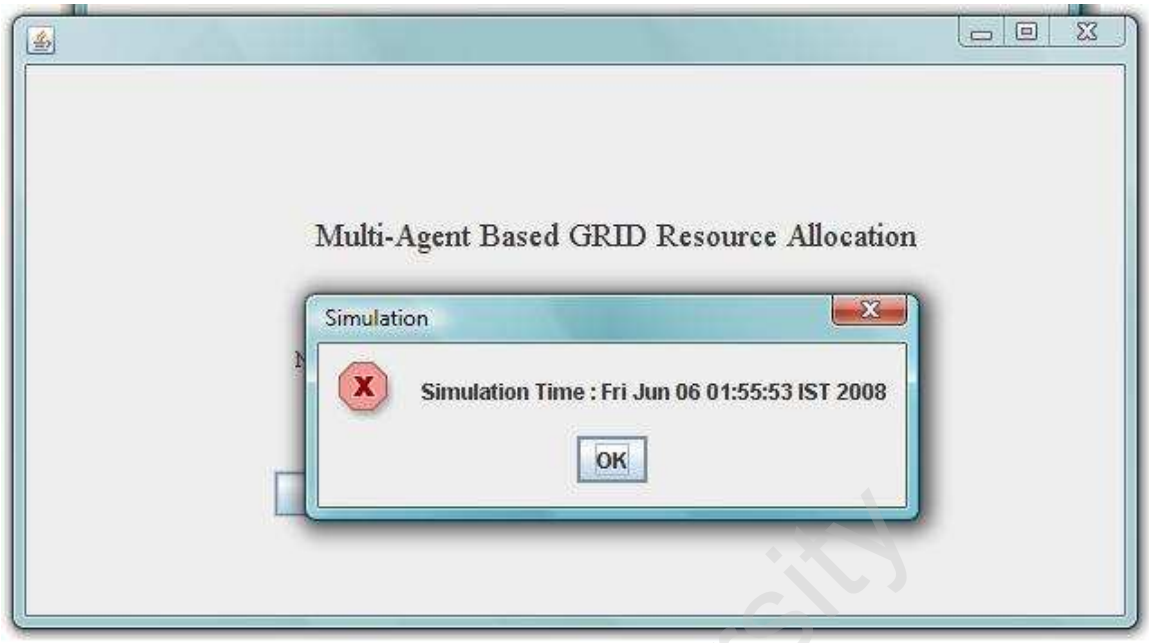
The application starts with taking the user input. The application asks for number of gridlets user required to run. The snapshot of the application is shown.



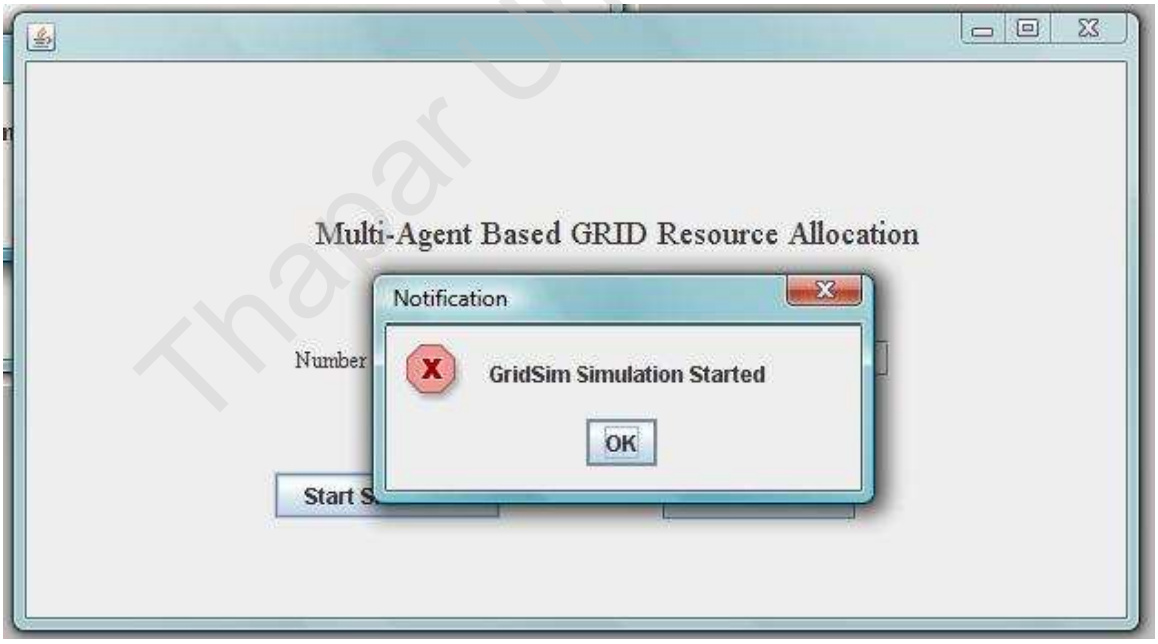
Snapshot 5.11 opening window with data entered

5.4.2 Notifications

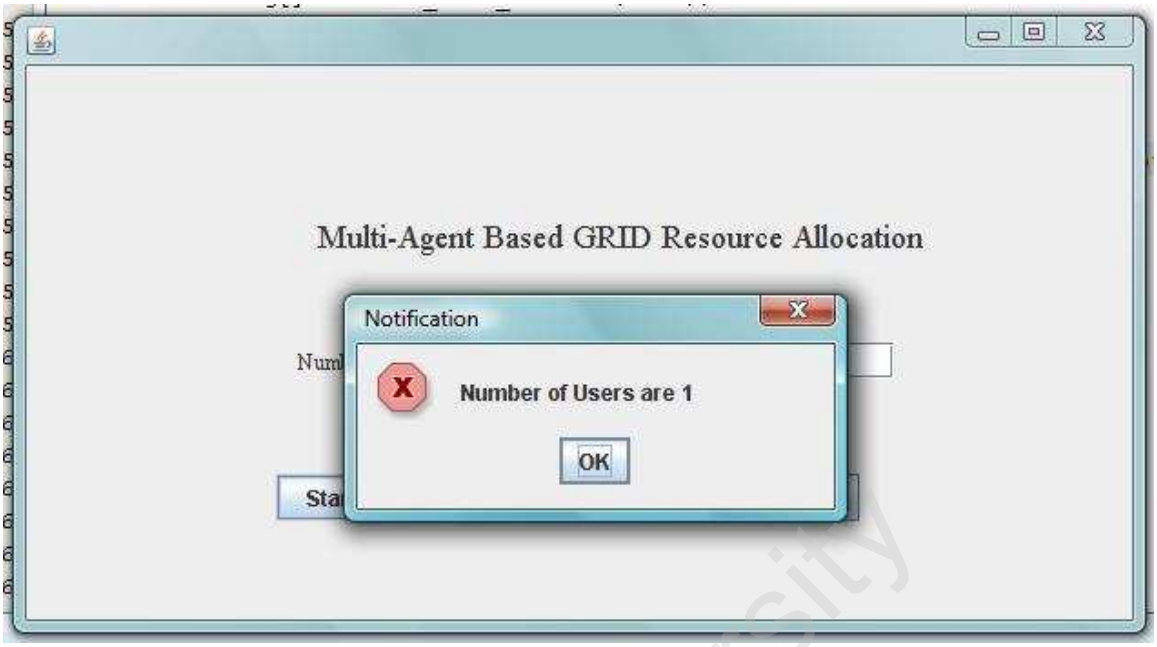
The next snapshots which will be shown consist of different notifications, given by GridSim.



Snapshot 5.12 Start Simulation Time



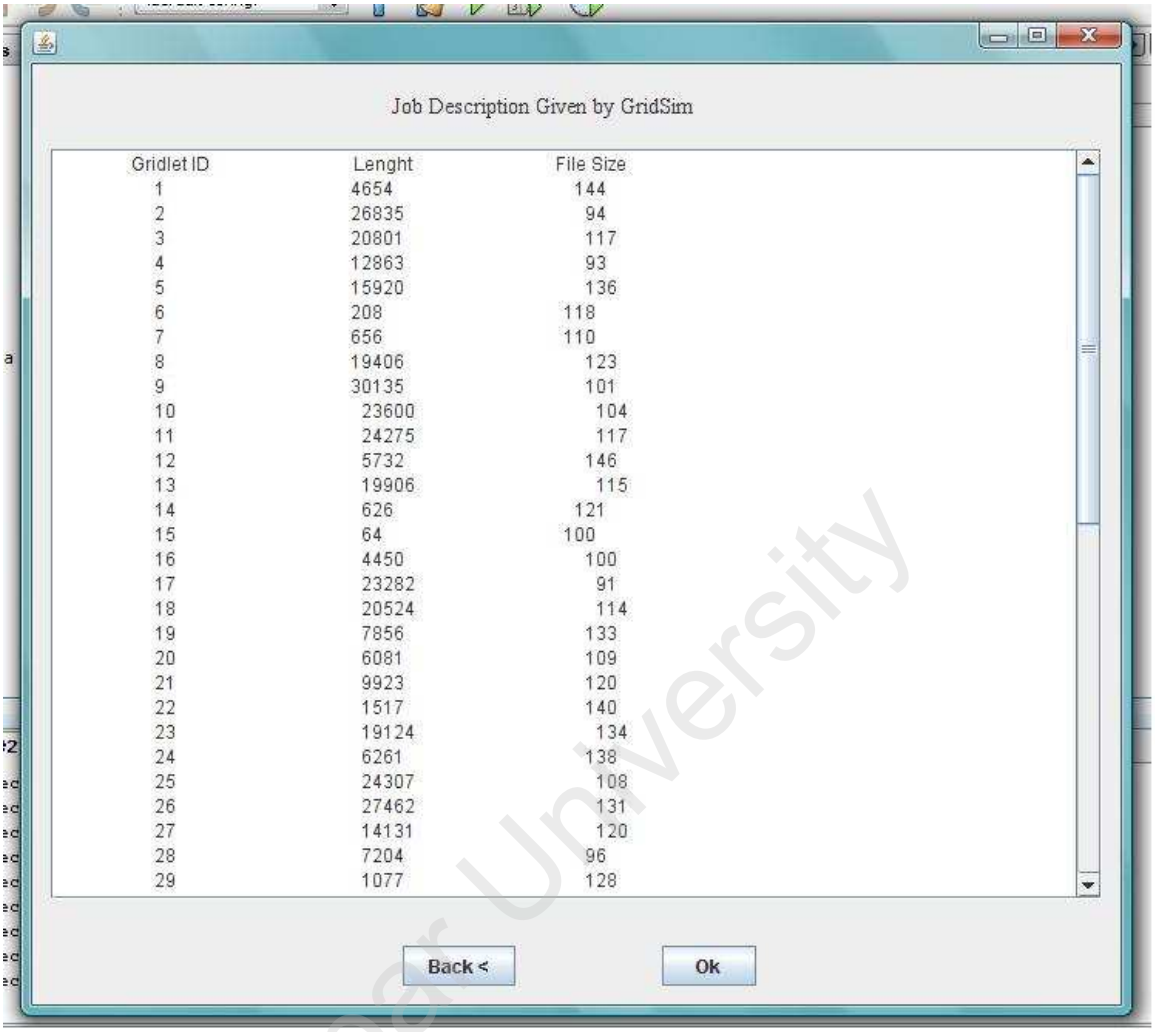
Snapshot 5.13 GridSim Simulation Started



Snapshot 5.14 Number of User 1

Snapshot 5.15 Number of User 3

The snapshot 5.14 and 5.15 shows the number users using the grid application at a particular time. The range has been set to 1-5, any number of agents can get active at a time.



Snapshot 5.16 Number of jobs created

The snapshot 5.16 shows the gridlets that are created on the request given by the grid user. As you can see form the image that every gridlet created has an unique ID, unique length, and unique file size. We send this gridlet to Broker to select the Agent and get the results back to the user. Snapshot 5.17, 5.18, 5.19 shows the results which agent is selected to execute the particular gridlet.

Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
Job 1:53.39	0.0	0.0	55.13	0.0
Job 2:286.50	0.0	284.08	0.0	0.0
Job 3:0.0	165.23	0.0	0.0	166.46
Job 4:135.81	0.0	136.68	0.0	138.23
Job 5:0.0	0.0	0.0	210.50	212.40
Job 6:0.0	3.82	0.0	0.0	2.89
Job 7:11.45	11.77	12.13	0.0	0.0
Job 8:0.0	0.0	258.78	257.08	0.0
Job 9:321.37	321.06	0.0	0.0	0.0
Job 10:313.14	313.44	0.0	0.0	313.69
Job 11:0.0	197.17	0.0	0.0	197.26
Job 12:62.52	0.0	0.0	61.66	0.0
Job 13:0.0	161.24	0.0	0.0	161.27
Job 14:0.0	0.0	4.39	6.63	0.0
Job 15:4.10	3.85	0.0	0.0	0.0
Job 16:60.04	58.48	0.0	0.0	0.0
Job 17:308.34	0.0	309.42	0.0	308.09
Job 18:0.0	221.35	223.12	0.0	222.39
Job 19:0.0	0.0	0.0	106.02	105.22
Job 20:51.71	52.81	0.0	0.0	50.76

Snapshot 5.17 Every agent giving their cost for the job

```
The selected Agent to Execute Gridlet_1 is Agent_1 with Cost of 53.
The selected Agent to Execute Gridlet_2 is Agent_3 with Cost of 284.08
The selected Agent to Execute Gridlet_3 is Agent_2 with Cost of 165.23
The selected Agent to Execute Gridlet_4 is Agent_1 with Cost of 135.81
The selected Agent to Execute Gridlet_5 is Agent_4 with Cost of 210.50
The selected Agent to Execute Gridlet_6 is Agent_5 with Cost of 2.89
The selected Agent to Execute Gridlet_7 is Agent_1 with Cost of 11.45
The selected Agent to Execute Gridlet_8 is Agent_4 with Cost of 257.08
The selected Agent to Execute Gridlet_9 is Agent_2 with Cost of 321.06
The selected Agent to Execute Gridlet_10 is Agent_1 with Cost of 313.14
The selected Agent to Execute Gridlet_11 is Agent_2 with Cost of 197.17
The selected Agent to Execute Gridlet_12 is Agent_4 with Cost of 61.66
The selected Agent to Execute Gridlet_13 is Agent_2 with Cost of 161.24
The selected Agent to Execute Gridlet_14 is Agent_3 with Cost of 4.39
The selected Agent to Execute Gridlet_15 is Agent_2 with Cost of 3.
The selected Agent to Execute Gridlet_16 is Agent_2 with Cost of 58.48
The selected Agent to Execute Gridlet_17 is Agent_5 with Cost of 308.09
The selected Agent to Execute Gridlet_18 is Agent_2 with Cost of 221.35
The selected Agent to Execute Gridlet_19 is Agent_5 with Cost of 105.22
The selected Agent to Execute Gridlet_20 is Agent_5 with Cost of 50.76
```

Snapshot 5.18 Selected Agent with their cost

Thus in the above two figures the results are shown that gives the agent selected along with it the cost per agent required to execute the job.

Chapter 6

Conclusion & Future Scope

In this thesis various terminologies related to Grid computing are studied. The resources available with the grid, advantages & disadvantages of using the grid computing are addressed. The relation of distributed computing with grid computing and the major difference between them are explained. Further explanation to resource allocation is given, its relation with the MultiAgent based Resource Allocation (MARA). The scope of the thesis lies in Resource allocation in MultiAgent based systems.

The thesis focuses on MARA, a new terminology in grid environment. The number of agents is used with whom the resource allocation takes place. The preferences for each agent are designed, based on which the job submission and resource allocation takes place. The proposed work focuses on the Auction based allocation procedure. English Auction based protocol is used for selection of agents. Certain Negotiation protocols related to MARA and Issues that are related to MARA are also addressed in this thesis.

The proposed algorithm for auction based protocol for resource allocation provides the more optimal and feasible way to allocate the resources. The Agent with available services and resources that are adaptable to the requirement of the job with less cost is selected to run the jobs. The Auction based protocol is implemented and successfully simulated on top of open source GridSim version 4.1 Toolkit together.

Finally the thesis concludes with, Resource Allocation with MARA gives more optimal and feasible results with a large pool of jobs. Thus MARA can be very and most useful for resource allocation.

Future Scope

The future scope for resource allocation is very large; certain points related to MARA can be:

- The foremost future work in MARA can be, the application is tested on GridSim toolkit 4.1. Next it can be implemented over number of the actual grid environment.
- The performance of the Algorithm can be improved by reducing the number of rounds of bidding. Making bidding more dynamic in the sense that, it should stop at a particular level.
- The matchmaking approach with MARA can be further explored for the Grid Resource Allocation.

References

[1] D.H.J Epema, *Distributed System*, second ed., TUDelft, 2003. < www.flazx.com >

[2] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson, *Impossibility of distributed consensus with one faulty process*, J. ACM 32 (1985), no. 2, 374.

[3] *IBM Red Book*, Dec 2005 Edition. http://www.redbooks.ibm.com

[4] M.L Liu, *Distributed computing principles and applications*, Addison-Wesley, may 2003

[5] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, *The physiology of the grid: An open grid services architecture for distributed systems integration*, (2002).

[6] Ian Foster, Carl Kesselman, and Steven Tuecke, *The anatomy of the Grid: Enabling scalable virtual organizations*, Lecture Notes in Computer Science 2150 (2001).

[7] Baker M., Buyya R., and Laforenza D., "Grids and Grid Technologies for Wide-Area Distributed Computing", Software: Practice and Experience, Dec 2002, vol. 32, no. 15, pp. 1437-1466.

[8] Franccois Grey, Matti Heikkurinen, Rosy Mondardini, *Grid Cafe: A Palace for Everybody to Learn About Grid*,
< http://Gridcafe.web.cern.ch/Gridcafe/Gridwork/architecure.html >

[9] Yanmin ZHU, *A Survey of Grid Scheduling*, Department of Computer Science Hong Kong University of Science and Technology.

[10] *Jean-Christophe Durand*, *Grid Computing: A Conceptual and Practical Study*, Universite de Lausanne, Nov 8, 2004

[11] Gangadhar Mahesh kandru, Economy Based Resource Allocation in Grid", 2006

[12] D. Kuokka and L. Harada, Matchmaking for information agents, Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 672{678}.

[13] Karl Czajkowski and Ian Foster and Nick Karonis and Carl Kesselman and Stuart Martin andWarren Smith and Steven Tuecke, *A Resource Management Architecture for Metacomputing Systems*, Lecture Notes in Computer Science 1459 (1998), 62{??}

[14] D. Kuokka and L. Harada, *Matchmaking for information agents*, Proceedings of the 14th International Joint Conference on Arti¯cial Intelligence, 1995, pp. 672{678}.

[15] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, *Economic models for resource management and scheduling in grid computing*, 2002.

[16] Yann Chevaleyre, LAMSADE, Universit´e Paris-Dauphine (France), "Issues in Multiagent Resource Allocation". Informatica: 1-2, May 12, 2005

[17] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 2nd edition, 2004

[18] T. W. Sandholm. Optimal winner determination algorithms. In P. Cramton, Y. Shoham, and R. Steinberg, editors, Combinatorial Auctions. MIT Press, 2006.

[19] Paul E. Dunne Department of Computer Science, University of Liverpool (UK) Ulle Endriss Department of Computing, Imperial College London (UK) Michel Lemaitre ONERA, Centre de Toulouse (France). Informatica: 1-2, May 12, 2005.

[20] Ulle Endriss, Department of Computing, Imperial College London (UK), "issues in Multiagent Resource Allocation". Informatica: 1-2, May 12, 2005.

[21] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. Journal of Finance, 16(1):8–37, 1961

[22] P.Cramton, Y. Shoham, and R. Steinberg, editors. Combinatorial Auctions. MIT Press, 2006. To appear. Informatica: 1-2, May 12, 2005.

[23] J. Kalagnanam and D. C. Parkes. Auctions, bidding and exchange design. In D. Simchi- Levi, S. D.Wu, and M. Shen, editors, Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era. Kluwer Academic Publishers, 2004.

[24] www.gridbus.org

[25] www.netbeans.org

[26] Buyya R (ed.). *High Performance Cluster Computing: Architectures and Systems*, vol. 1. Prentice-Hall: Englewood Cliffs, NJ, 1999.

[27] Oram A (ed.). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* O'Reilly, 2001.

[28] Rajkumar Buyya and Manzur 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, Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[29] Aridor Y, Factor M, Teperman A. cJVM: A single system image of a JVM on a cluster. *Proceedings 29th International Conference on Parallel Processing (ICPP 99)*, Fukushima, Japan, September 1999. IEEE Computer Society Press, 1999.

[30] Howell F, McNab R. SimJava: A discrete event simulation package for Java with applications in computer systems modelling. *Proceedings of the 1st International Conference on Web-based Modelling and Simulation*, San Diego, CA. Society for Computer Simulation, 1998.

List of Papers Published / Accepted

1. Seema Bawa, Kabir Oberoi, "MultiAgent Based Resource Allocation In Grid Environment", **National Conference on Research & Development in Hardware & Systems (CSI-RDHS 2008),** June 20-21, 2008 at Science City, Kolkata, INDIA. [*Accepted].