Market Economy Based Resource Allocation in Grids

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

Master of Technology

In

Information Technology

By

Sai Rahul Reddy P
[Roll No. 04IT6011]

Under the supervision of
Dr. Arobinda Gupta

School of Information Technology
Indian Institute of Technology, Kharagpur
India
May 2006

School of Information Technology
Indian Institute of Technology
Kharagpur-721302

Certificate

This is to certify that the Thesis titled "**Market Economy Based Resource Allocation in Grids**", submitted by Sai Rahul Reddy P, to the School of Information Technology, in partial fulfillment for the award of the degree of **Master of Technology (Information Technology)** is a bona-fide record of work carried out by him under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of the institute and, in my opinion, has reached the standard needed for submission.

Dr. Arobinda Gupta
School of Information Technology &
Department of Computer Science and Engineering
I.I.T. Kharagpur - 721302

ABSTRACT

In this thesis, we study several auction based resource selection policies for users in grid, which assist them in choosing the resource according to user preference. While choosing the resource these policies try to optimize parameters like average turnaround time, average budget per job, and number of jobs finished within deadline according to user preference. First we proposed simple TimeOptimized and BudgetOptimized policies which improve only one parameter i.e. either average turnaround time or average budget per job. We compared these policies with a Random policy which selects resources randomly. Later we improved these algorithms and proposed NewTimeOptimized and NewBudgetOptimized policies which consider the success rate also. We next presented a policy that considers the relative preferences of the user for the different parameters in selecting a resource. Finally we proposed a history based policy that tries to improve above parameters by considering the previous bids. We used GridSim simulation framework to evaluate our policies.

ACKNOWLEDGMENTS

It brings me immense pleasure to express my deepest sense of gratitude to my guide **Dr. Arobinda Gupta** for his expert guidance and support throughout the project work. His suggestions and invaluable ideas provided the platform to the whole project work. In spite of his extremely busy schedule, I have always found him accessible for suggestions and discussions. I look at him with great respect for his profound knowledge and relentless pursuit for perfection. His ever-encouraging attitude and help has been immensely valuable.

Nothing would have been possible without the support of my family members, who have been backing me up throughout my life. I wish to convey my sincere thanks to my parents; **Partha Saradhi Reddy P**, **Lakshmi Thulasi P**, and my sister Sai Prasanthi P who have given me an endless support and love and who also provided me with the opportunity to reach this far with my studies.

I would like to express my sincere thanks to **Dr. Shamik Sural** for providing the printing facility. I would like to express my thanks to all faculty members, my classmates and all my friends in IIT Kharagpur for their support.

Sai Rahul Reddy P,
School Of Information Technology,
Indian Institute of Technology,
Kharagpur - 721302, INDIA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Grid computing has emerged as a promising next generation collaborative problem solving platform for industry, science, and engineering. Grid computing is defined as coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [19]. The sharing ranges from simple file transfer to direct access to computers, software, data, and other network accessible resources. At the heart of the grid is the ability to discover, allocate and negotiate the use of these resources. Grid enables this sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations [3]. Grid computing is generally used for problems with large scale collaboration and huge computational and/or data storage requirement. The applications of grid includes large scale simulations in astrophysics, climate modeling, modeling for drug design, high energy physics, infrastructure for multiplayer games etc [4, 21, 22].

The main characteristics of a grid are [3]:

- 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 access different resources.

- Scalability: A grid might grow from few integrated resources to millions. The performance might be degraded as the size of the grid increases. The applications that require a large number of geographically located resources must be designed to be latency and bandwidth tolerant.

- Adaptivity: Grid should be resilient to failure of individual nodes. As it contains a large number of nodes, the probability of failure of nodes will be high. So the applications and resource brokers behave should adapt dynamically and use the available resources and services efficiently and effectively.

1.1. Overview of a Grid

In this section, we will explain about the various entities in a grid computing environment and interaction between them. The main entities in a grid are user, resource broker, information service, and resources. The interactions between these entities is shown in Figure 1. For each user, there will be one resource broker. Users in grid will submit their program to the resource broker. The resource broker in turn will find out appropriate resources and submit the program to the resource. Here a resource may be computational resource or storage resource or network resource or any device that is able to participate in the grid. Resources at a minimum should implement enquiry mechanisms that permit discovery of their structure, state and capability [19].

The resource broker contacts an *Information Service* to find an appropriate resource. The Information Service maintains complete information of all resources (like its capability, status, contact information etc) available in the grid. The resource broker will select one among them and submit the job to it. After authentication, the remote resource will execute the user program. In executing the user program, it may in turn need another resource to complete

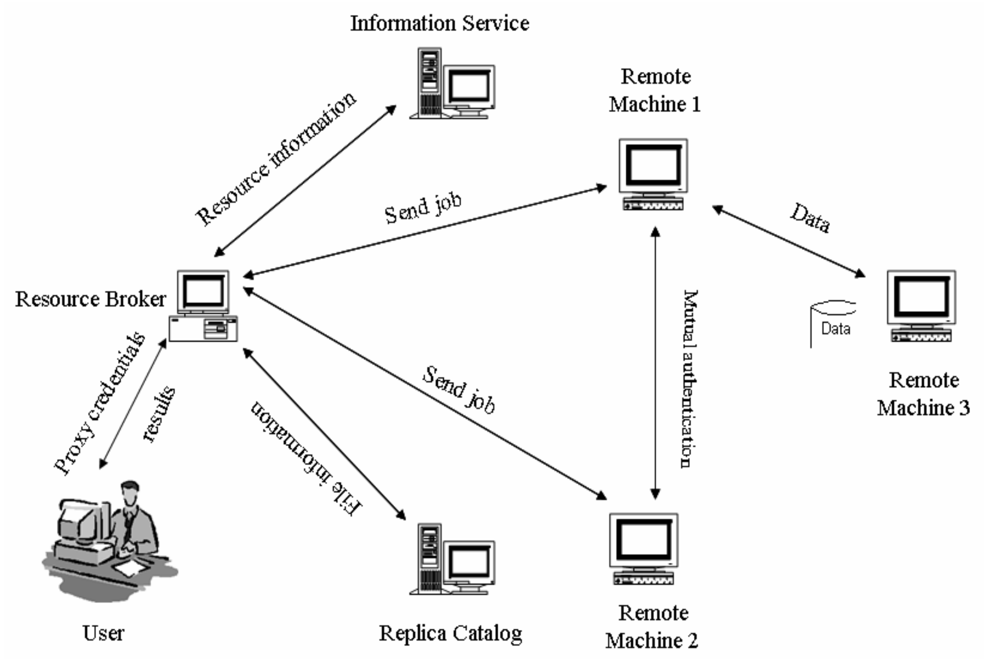

Figure1: Working of the grid

the job (like, it may have to access data from other machine or it may need to make the execution faster). If the user has appropriate credentials to access this new resource, then the remote machine (currently using machine) will get authenticated on behalf of the user (single sign-on) and it will use this new resource to complete the job.

A resource broker may need to access, *Replica Catalog* to locate the data. The replica management system controls where and when copies of files are created, and provides information about where files are located. In other words a replica management system maintains a mapping between logical names for files and collections to one or more physical locations.

1.2. Market Economics

For exploiting the full potential of a distributed system consisting of resources with comparable but different capability and availability, a mechanism is required to gather and compare such information for various resources in the system and assign each job to the most appropriate resource [34]. Viewing the resources as suppliers and the users as consumers of computing services, markets for computing services/resources have been examined as one of the most promising mechanisms for global scheduling [34]. Framing the resource allocation problem in economic terms is attractive for several reasons [48].

- Resource usage is not free. Initially the main motivation of building grids is to support research. To make grids successful commercially, resource owners and users should get adequate rewards. The main aim of users will be to execute the maximum number of jobs, satisfying QOS requirements etc. Similarly the main aim of resource owners will be to make profit out of the resources.

- The dynamics of grid performance are difficult to model. By formulating gird resource usage in market terms, we are able to apply analytical research from economics for understanding of the behavior of grids.

- Market formulation carries with it an inherent notion of relative worth which can be used to quantify the cost to benefit ratio for both grid users and resource owners.

The first step to use market economics is to define the market mechanism to be used. Here market is where goods and services are bought and sold and market mechanism is the process by which the market solves the resource allocation problem, especially deciding how much goods or service should be produced, deciding the price of goods and other such problems [51]. Broadly, market mechanisms can be categorized into one of three types.

- **Commodity markets:** In this type of market model, we treat different resources like computers, disk storage, bandwidth, and applications as commodity goods and we purchase from the suppliers.

We can pay the cost of the resources in different ways. Pricing schemes in commodity market model can be based on flat fee, usage duration, subscription, and demand and supply based. In the flat fee scheme, buyers will pay a fixed amount for a certain period irrespective of the service quality. The second scheme is based on the usage duration. If the resource is used for one hour then the buyer will pay for that one hour only. In the third scheme, i.e. subscription based, the user pays a fixed price for a certain duration. It is thus a more generalized form of the flat fee model. In the final scheme, the prices will change dynamically based on the supply and demand of the product. The disadvantage of the first three pricing schemes is that they do not exploit the demand for the resources. If there is a high demand for a resource, then it is not desired to sell the resource at a lower price. Similarly, if the demand for a resource is less, then decreasing the resource price may attract new users.

- **Tendering/Contract Net:** In this model, the resource broker will ask for bids from the sellers. A buyer will send the specification of the process (expected run time, resource requirements etc.) to the potential providers and the interested resource providers will participate in the bidding process. The resource broker will evaluate the bids and will give the contract to the most appropriate one.

- **Auctions:** In this model, a resource provider accepts bids from the resource broker for the resource. There are two types of bids, open and closed bids. In open bidding, participants will know the bid amount of the other players. In closed bids, participants will not know the bid amount of the other players. There will be a period of time in which the resource providers will accept bids and after the end of the bidding period the resource provider will evaluate the user bids and will give the resource to the appropriate resource broker. Auctions can be of several types such as first price sealed bid auction, vickrey auction, english auction, double auction etc. More details on auctions can be found in [5].

The money exchanged between a user and a resource provider may be real money or virtual money. As we said earlier, here we are using markets as a controlling mechanism for allocating resources to users. The demand and supply of resources determine the prices and in turn controls the allocation. If demand for some resources is high, then price of the resource will also be higher and very few will be able to afford that resource.

1.3. Motivation of this Work

In our work, we focus on auctions as the market mechanism to use in grids. The advantages of auction over other market models are:

- The auction model supports one-to-many negotiation between a service provider and many consumers, and reduces negotiation to a single value.

- Auctions require little of global price information and are easy to implement in grid settings [5].

- Unlike commodity market model, auctions are completely decentralized.

There exist several studies on applying auctions to solve resource allocation problem in grids [9, 25, 28, 29, 48, 49]. Most of the studies in auctions assume that the resources are homogenous. For example, most of them assumes that all resources have the same speed and cost. In such a case, choosing resources is trivial as choosing any random resource will serve the purpose. But in a real world environment, resources will be heterogeneous. When we consider heterogeneous resources different parameters like resource architecture, resource speed, available memory, price of resource, bandwidth charges etc will come into picture. The choice of a proper resource for an application will thus be based on various parameters. Moreover, some of these parameters may be interdependent and trying to optimize one can affect the other adversely. For example, if we choose resources with high speed, then the cost will be higher. So, in such cases, there should be some mechanism to choose the resources according to the user's requirements which should consider the user preference as well as chances of winning in the auction.

1.4. Problem Statement

In our work, we considered resources with different capabilities (speeds) and prices. A set of users, each with a set of jobs, wish to use these resources. Each job has a deadline and an allowed maximum budget. We considered two parameters; time and budget. The problem is to define resource allocation policies that allocate resources to the jobs while increasing the number of jobs finishing within their deadline and decreasing the average turnaround time and the average budget spent for these jobs. The allocation policies should be able to take into consideration user preferences on which parameter to optimize more. We introduce several algorithms that optimize time and budget to different extents.

1.5. Contributions

In this thesis we introduced policies for users to choose resources in a grid environment using auction. The main contributions of this work can be summarized as follows:

1. We first presented three policies for resource allocation - Random, TimeOptimized and BudgetOptimized policies. In Random policy, a user chooses resources randomly. In TimeOptimized, a user will select resources based on the completion time of job and in BudgetOptimized,

user select resources based on its cost. TimeOptimized policy tries to optimize the average turnaround time of jobs and BudgetOptimized tries to optimize the average budget spent per job. If every user in grid uses the same policy then there will be contention for high speed/low cost resources and because of this, some jobs may loose the deadline. Thus the number of jobs finishing within deadline in TimeOptimized and BudgetOptimized policies is less when compared with the Random policy. Random selection maximizes chances of winning but the turnaround time is higher than TimeOptimized policy and average budget spent per job is higher than BudgetOptimized policy. We then proposed two new policies. The first one, NewTimeOptimized policy, tries to minimize average turnaround time while increasing the number of jobs finishing within deadline. The second one NewBudgetOptimized policy, tries to minimize the budget spent per job while increasing the number of jobs finishing within deadline

2. We next introduced a resource selection policy that tries to optimize both the above parameters; average turnaround time and average budget spent per job based on user preference. We have used Analytic Hierarchy Process (*AHP*) technique to model the user preference.

3. Finally we introduced policy where the user will give preference in terms of success rate. The success rate is defined as the number of jobs finishing within deadline. Initially we start with NewTimeOptimized policy and when success rate is reaches the user given value, we will shift to NewBudgetOptimized policy. In NewBudgetOptimized we do not use our complete budget for a job to bid. So first we are using our complete amount for bidding in the initial stage, and after getting sufficient success rate we are shifting to NewBudgetOptimized policy. This policy effectively tries to reduce the average budget spent per job while maintaining the user requirements.

1.6. Organization of the Thesis

The thesis is organized into the following chapters. In Chapter 2, we briefly explain the various auction mechanisms used in grid setting and then we discuss related work in this field. In Chapter 3, we discuss the overall system model we used in our simulations. After that we discuss about the simulator we used. In Chapter 4, we discuss Random, TimeOptimized, and BudgetOptimized policies and their results. In Chapter 5, we presented the NewTimeOptimized and NewBudgetOptimized policies and their results. In Chapter 6, we discuss a user preference based allocation method and its results. In Chapter 7, we present a history based policy and its results. In Chapter 8, we conclude the thesis and discuss the future work.

Chapter 2

RELATED WORK

The market mechanism used in this thesis is auctions. In this chapter, we first discuss the auction mechanism in detail and then describe the work done using auction in grids.

2.1. Different Auction Mechanisms

There are different variations of auctions. Auctions mainly differ in two aspects, whether they are open cry or closed, and whether they are ascending or descending auctions. In open cry auctions all the participants know the other participants bid information. In closed auctions, participants do not have access to other participant's bid information. The ascending auctions start with a low price and will go higher and higher until no one bids. Descending auctions start with a high price and will go lower and lower until one accepts the bid. Based on the above criterion auctions can be divided into the following types.

- **English Auction:** The Auctioneer will start the auction with the reserve price (lowest acceptable amount) and takes larger and larger bids until no one will increase the amount. The auctioneer will give the resource to the highest bidder.

- **Sealed bid:** The Auctioneer accepts bids from users in which the players will not know the other player's bid amount. At the end of the auction period, the auctioneer opens the bids and gives the resource to the highest bidder.

- **Vickrey Auction:** In Vickrey or Second price sealed bid auction the bidders will bid the amount without knowing the other bidders' amount. The bidder who bids the highest amount will get the resource but he/she will pay the price of the second highest bid amount.

- **Dutch Auction:** It is similar to English Auction but the bidding process starts with the highest amount instead of the lowest amount. The auctioneer will continuously decrease the amount. The bidder who can pay the current bidding amount will get the resource.

- **Double Auction:** This type of auction is common in stock exchanges. In this type of auction sellers' offer are called asks and the bidders' amount are called bids. The restriction is that the seller must ask a price that is less than the current ask and the bidder must bid an amount higher than the current highest bid. When a match occurs between asks and bids, the transaction is committed. Another type of double auction is Clearing House Auction in which the bidders will submit their bids and the sellers will submit asks. Once submitted, asks are sorted in ascending order and bids are sorted in descending order. The price is the average of the lowest ask and the highest bid offer. This type of auction is normally considered fair. The only difference between normal double auction and clearing house auction is that in normal auction, the transaction is committed immediately after a match, and in clearing house double auction, it will be after some specific time.

In our work, we concentrated mainly on sealed bid auction. One should choose a mechanism for which truthfully revealing one's true willingness to pay is a dominant strategy. A mechanism of this sort is called direct mechanism or strategy proof mechanism [44]. Vickrey auction is such a mechanism but vickrey auction will not be of much help in a grid setting. In a grid environment, the users will participate again and again in repeated auctions and so the users can behave strategically. To simplify the problem, we are not considering strategic users in our work.

The second reason why vickrey auction is not much helpful in our settings can be explained using the revenue equivalence theorem [30]. The revenue equivalence theorem states that when bidders are risk-neutral and have independent private values (but it does not hold for common values with risk averse bidders), any auction format will on an average generate same expected revenue. However, revenue equivalence breaks down when bidders are risk-averse. Here independent private values means that each bidder knows how much it values the objects for sale, but its value is private to itself [30]. Common values means that the actual value is the same for every one but bidders have different private information about the actual value [30]. For example, if resources are not permanent in grid then before bidding, users have to consider the participation time of resources in grid as well. In that case all the users value the resource the same but each user estimates the participation time differently. The bidder would change his/her estimate of the value if he/she learnt another bidder's estimate. This is in contrast to the private value case in which his value would be unaffected by learning other bidders information.

Describing how to design auctions efficiently is out of the scope of this document. Economists use game theory to model participant's interaction and the subject that deals with this is mechanism design (also called implementation theory). More information on mechanism design in auctions can be found in [30, 44].

2.2. Related Work

Research work in the area of market economics in grid computing can be classified into three areas.

The first area of work examines different market mechanisms for grid environments. Here a mechanism may be auction, commodity market etc. The mechanism that is applied should encourage users and resource owners to participate in the grid. There exists several studies that compare different market mechanisms for grid environment [9, 25, 29, 48]. Here we will discuss few of them briefly. Wolski et al. [48] compares commodity market model and auctions with respect to price stability, market equilibrium, application efficiency and resource efficiency. He concludes that commodity market model is better than auctions with respect to the above parameters. Wolski has used Smale's technique [48] for finding unit price in commodity market. The problem with this technique is that there should be some cooperation between the service providers or some regulatory authority which decides the unit price (its job is to find the supply and demand of the service providers and calculate the unit price accordingly). This is not practical in real world grid environments. Grosu and Das [25] compares first price, vickrey, and double auction with respect to user payments, resource profits, payment structure, and resource utilization. They conclude that first price auction is better from resource perspective, vickrey is better from user's perspective, and double auction is better for both. The problem with double auction is that again there should be some cooperation between resource providers. Kant and Grosu [29] compare different double auction protocols with respect to the above parameters.

The second area of work examines the scheduling strategies for resource providers. Xiao et al. [49] deals mainly with scheduling of accepted jobs at the server side. It uses tender/contract-net economic model. When a resource receives notification of a new job, the resource has to decide whether to accept the job or not. It might have sent a bid request to some other user and is waiting for the response, and in this case, if it sends again it may eventually get both jobs and one of them may miss the deadline. To control such behavior they introduced penalty for resource providers if they did not meet the deadline and thus prevents them from accepting more jobs than they can handle. This is controlled by conservative degree (*CD*). *CD* = 0 means it is aggressive and accepts all jobs. *CD* = 1 means it is conservative. They compared conservative degree with failure rate and deadline miss rate. At low system load and under low *CD* there are no deadline misses and job fails. Deadline miss rate is increased when the system load is increased but failure rate is not increased. At high *CD* deadline miss rate is zero but failure rate increases with increasing load. Ernemann et al. [15] also deal with scheduling the jobs at the server end that maximizes the given utility function. Here utility is like minimizing startup time etc. Kale et al. [28] uses tender/contract-net economic model. It also investigates scheduling algorithms that will be best suited for resource providers. It compares Gantt chart scheduling and best fit strategy with respect to loadfactor vs revenue gained, loadfactor vs percentage work done, loadfactor vs percentage utilization, and loadfactor vs percentage of rejected jobs.

The third area of work has attempted to find resource selection policies for users. Buyya et al. proposed Nimrod-G [6] which supports several economic models like commodity market, spot market, and contract net. It implements two resource selection policies for the above market models, time optimization, and budget optimization policies. The problem with this approach is they assumed that one centralized agent will do the scheduling for all the users. In their work they have not applied those policies to auctions. In auctions, we can not directly use those policies.

In our model we assumed that for each job there will be certain amount allocated to it and for each job there will be a deadline associated with it. The job should finish its execution within its deadline and budget. The work in [25, 29] considered resources with different capabilities but they selected resources randomly for bidding. The problem with random policy is that the resource capabilities are not considered for bidding. Hence the average turnaround time and the average budget spent per job both increases.

In the next chapter, we discuss the overall system model we used in our simulations. After that we discuss about the simulator we used.

Chapter 3

SYSTEM MODEL

In this chapter we first explain the system model and then we explain the simulator we used.

3.1. System Model

The grid computational environment consists of resource consumers or users and resource providers. Resource consumers have jobs to be done and are willing to pay for it. Resource providers have computational resources and are willing to rent them for profit. Scheduling enables the interaction between the two parties and maps jobs to resources properly [48]. We used sealed bid auction as our market mechanism.

Each resource consumer or user has its corresponding resource broker and submit their jobs to the resource broker. A resource broker will take care of searching for suitable resource providers and submitting the job to a resource provider. Let U_1, U_2, U_3 … U_N be the users participating in the grid and J_1, J_2, J_3 … J_K be the corresponding jobs for each user. Each job specification J_i includes job length, deadline, and budget. The jobs have to be completed within its deadline and its cost of execution should not exceed its allocated budget. The job length is specified in millions of instructions (MI). The deadline includes the time spent on the auctions also.

A resource provider executes jobs for resource consumers and charges them for usage of resource. Let R_1, R_2 … R_m be the resources participating in the grid. Each resource R_i is modeled by processor speed and unit price. The capability of resources is expressed in terms of millions of instructions the resource can process in one second (MIPS). The unit price is the amount a user pays for one second usage of the resource. Here it specifies the minimum price the resource R_i accepts. In our work we considered resources with single processor. Each resource R_i will conduct auction A_i. Users who want to use the resource have to participate in the auction. In the rest of the thesis, the speed of the resource i is referred as R_i.*speed*, resource usage start time is referred as R_i.*resource_usage_start_time*, and unit price of the resource is referred as R_i.*price* .

The main job of a resource broker is to find an appropriate resource according to user policy and to bid for that resource. Let Rb_1, Rb_2 … Rb_N be the resource brokers for the users U_1, U_2 … U_N respectively. A user will specify which resource selection policy resource broker has to use and the resource broker in turn select resources accordingly. In the rest of this thesis, we have not differentiated much between the resource broker and the user and we have used resource broker and user interchangeably. The details of these policies will be explained in the next chapter.

Grid Information Service (*GIS*) contains complete information about current auctions. Each auction description A_i includes the resource provider id, auction number, starting time of resource usage, auction end time, reserve price and capability of resource. It does not include resource usage end time; it depends on the job it accepts in the auction. The resource broker will first contact the GIS for auction information. Resource providers will periodically update their auction information in GIS. It uses a soft state protocol, meaning that the GIS will not query resource providers for the latest information. It is the job of a resource provider to provide the latest information about current auctions to GIS.

Each resource provider will conduct sealed-bid auction and accept the bids until the end of auction period. Here the bidding amount is in terms of cost/sec. The bidding amount should be greater than the reserved price for that resource. At the end of the auction, the resource provider will open the bids and inform the resource brokers whether they won in the auction or not. The maximum bid amount is not revealed to others. This is to prevent resource brokers from behaving strategically. A resource broker may learn the bid amount to bid by participating in repeated auctions, but we are not considering it here. If no one participated in the bidding by the end of the auction, the above process will be repeated.

For TimeOptimized policy, the resource broker uses its complete amount allocated to the job for bidding but for BudgetOptimized policy, it calculates the penalty in execution time by choosing low cost resources. It reduces the bid money proportional to that. The choosing of a particular auction from set of auctions for bidding will be explained in the next chapter. After the end of the current auction, the resource providers will start new auctions for the next available time slot. The starting time of the next usage is changed in accordance with the current accepted job. The complete interaction between users and resource providers is shown in Figure 2.

The complete interaction between users and resource providers can be summarized as follows. Whenever a job is available to a user, it will submit the complete job specification to its resource broker. The resource broker will query GIS for the current auction information. After getting information from GIS, according to predefined policy specified by the user, the resource broker will choose one provider and bid for that resource. At the end of the auction, the resource provider will inform whether it won in the auction or not. If it won, it will submit the current job to the resource provider. If it did not win then it again repeats the whole process. At the end of the execution of the current job, the resource provider returns the result to the resource broker.

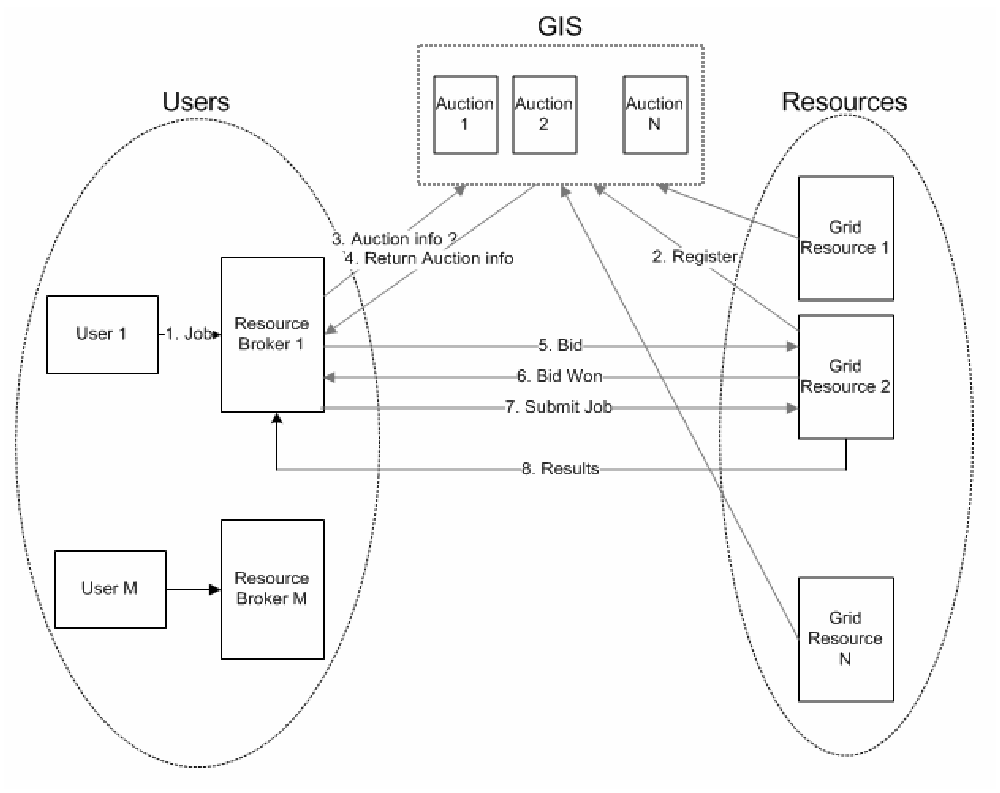

Figure 2: Interaction between users and resource providers

3.2. GridSim

We used *GridSim* [7] for evaluating the proposed user selection policies. GridSim is a java based discrete event grid simulation toolkit. GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domain distributed computing systems such as clusters and grids [7]. Other features of GridSim include advance reservation capability, network topology consideration capability, background network traffic functionality, and support for different time zones.

GridSim uses s*imjava*. Simjava is a discrete event simulation package for java. The main classes in GridSim are *GridSim, GridResource, AllocPoilicy, GridInformationService,* and *Gridlet.* All the entities in the GridSim are derived from the GridSim class. The GridSim class provides methods for sending and receiving messages between entities, managing and accessing handles to various GridSim core entities and recording statistics [7]. The GridResource class is derived from the GridSim class and act as a grid resource entity. The GridResource class can be used to create machines with single processor to multi processor machines and clusters as well. By default the GridResource class provides two scheduling algorithms, *time share* and *space shared* algorithms. We have to extend the AllocPolicy class to provide our own scheduling algorithm for scheduling the jobs at the resource end. The GridInformationService class is a GridSim entity that provides resource registry, indexing and discovery services. The Gridlet class is a job package that stores complete information about the job like job length, job deadline, etc.

The communication between the entities in GridSim is through messages. All the entities in GridSim are java threads. Simjava maintains the queue of messages and delivers the message to appropriate entities at the right time. Users and resource brokers are derived from the GridSim class. A user will submit jobs to a resource broker and the resource broker will participate in the auction. Before accepting the job, the user has to participate in the auction conducted by the resource. Each resource will conduct an auction on its own. To simulate that, in GridSim, we extended GridResource class to provide the auction capability into it. The interaction between the user and the resource, before submitting a job, can be visualized as message exchange between them. So to provide auction capability into GridResource class we added message handlers into it. Similarly we modified the resource broker class.

In the next chapter, we discuss Random, TimeOptimized, and BudgetOptimized policies and their results.

Chapter 4

RESOURCE SELECTION POLICIES

In this chapter, we first implement three simple policies for resource allocation - Random, TimeOptimized and BudgetOptimized policies. The user will specifies a policy for its resource broker and it will choose the resource according to that policy.

4.1. Resource Selection Policies

- Random policy: In Random policy, a user randomly chooses one resource for bidding that can complete the job within the deadline and budget allocated for the job. The pseudocode for Random policy is shown in Figure 3.

```
S = { NULL };

// Find set of resources who can complete the job within deadline and
budget
for (i=0; i<n; i++){

    // minimum time required to execute the job
    exec_time = job_length / $R_i$.speed;

    // completion time on Resource i
    completion_time = $R_i$.resource_usage_start_time + exec_time;
    if ((budget >=($R_i$.price * exec_time )) AND (completion_time<=deadline) ){
        S = S ∪ $R_i$;
    }
}
select randomly one resource $R_i$ from S;
```

Figure 3: Algorithm for Random policy

- TimeOptimized policy: In TimeOptimized, a user will always bid for a resource that can complete the job the earliest within the deadline and budget allocated for it. It uses the whole budget allocated to the job for bidding for the selected resource. If it fails in the current auction, it then chooses the next resource that can complete the job within deadline and bids for it. This continues until the job either finds a resource or misses the deadline. The pseudocode for TimeOptimized policy is shown in Figure 4.

```
S = { NULL };

// Find set of resources who can complete the job within deadline and
budget
for (i=0; i<n; i++){

    //  minimum time required to execute the job
    exec_time = job_length / $R_i$.speed;

    // completion time on Resource i
    completion_time = $R_i$.resource_usage_start_time + exec_time;
    if ((budget >=($R_i$.price * exec_time )) AND (completion_time<=deadline) ){
        S = S ∪ $R_i$;
    }
}
sort S by completion time;
select $R_1$ from S;
```

Figure 4: Algorithm for TimeOptimized policy

- BudgetOptimized policy: In BudgetOptimized case, a user will always bid for a resource that costs less. In BudgetOptimized case it will not use its whole amount allocated to the job for bidding. Instead, it calculates the penalty and it will reduce the allocated amount in proportion to the penalty. Here penalty is the degradation in the performance a user is getting by choosing a low speed resource. The penalty is high for low speed resource and it is very low for high speed resource. So it bids less for a low speed resource and it bids higher for a high speed resource. If it fails in the current auction, it then chooses the next high cost resource which can complete the job within deadline and bids for it. The pseudocode for BudgetOptimized policy is shown in Figure 5.

```
$S$ = { NULL };

// Find set of resources who can complete the job within deadline and budget
for ($i$=0; $i<n$; $i$++){

    //  minimum time required to execute the job
    exec_time = job_length / $R_i$.speed;

    // completion time on Resource $i$
    completion_time = $R_i$.resource_usage_start_time + exec_time;
    if ((budget >=($R_i$.price*exec_time)) AND (completion_time<=deadline) ){
        $S = S \cup R_i$;
    }
}
sort $S$ by price;

// $R_1$ will be the resource with lowest budget
select $R_1$ from $S$;

// max_speed is the maximum speed of the resource the user can bid
penalty = $R_1$.speed / max_speed;

// minimum amount needed to complete the job
min_amount_needed = (job_length/$R_1$.speed)*$R_1$.price;
bid_amount = budget – (budget –min_amount_needed)*penalty ;
use bid_amount to bid for $R_1$;
```

Figure 5: Algorithm for BudgetOptimized policy

4.2. Simulation Results

In this section we have provided simulation based evaluation of the policies we explained in the previous section.

4.2.1. Experimental Methodology

The simulated grid environment consists of 15 resources and 10 users. Resources have different processing speeds and reserve prices as given in Table 1. The processing rates are within the range [400, 2000], which includes low speed to high speed resources and characterizes real grid environment. The reserve prices for these machines are in the range [2, 18] and chosen such a way that price per MI is increasing when we go from low speed resource to high speed resource. The increase in the amount per MI is the premium paid to the resource for executing the job faster. If we have given same price per MI for all resources, then the user always chooses high speed resources only. We assumed that each machine will execute one job at a time. After completion of the present job it will again start new auction. There are a total of 50 jobs for each user.

Incoming jobs for each user will come according to Poisson distribution with mean □. For all experiments we have kept □ constant at 0.01. If we choose high □ then the jobs fail because of high load so we have chosen a low □. A job should not miss its deadline. The job deadline includes auction participating time, execution time of job, and waiting time at the resource end. For each job, deadline is set according to the following expression.

$$J_i.\textit{deadline} = E_{ij} + \mathrm{Rand}(E_i) + □ .$$

where $J_i.\textit{deadline}$ is the deadline for job i, E_{ij} is the execution time of J_i on R_j where R_j is the slowest processor available in the grid. Execution time is calculated as job length / MIPS of processor, $\mathrm{Rand}(E_i)$ is a random value between 1 and E_{ij}, and □ is positive constant. This constant is added to alleviate the effect of time spent on auctions. For all our experiments we kept □ constant at 30. This value is equal to the default auction time.

The budget for each job is distributed uniformly over the interval [$□_1$, $□_2$]. We have taken this approach from [12]. The lower limit $□_1$ of jobs budget interval is given by the product of the lowest computational time of a job and lowest reservation price of a resource while the upper limit $□_2$ is given by the product of highest computational time of a job and the highest reservation price of a resource.

In TimeOptimized policy, the user uses his entire amount for bidding but in BudgetOptimized policy, the user does not use his entire amount. Instead he calculates the penalty in performance he is getting by choosing the low speed resource instead of high speed ones and reduces the money in proportion to the amount allocated to it. The bidding amount for a job is calculated as shown in Figure 5.

4.2.2. TimeOptimized and BudgetOptimized Policies

In the first experiment, we implemented Random, TimeOptimized and BudgetOptimized policies. The job length in this experiment varies from 10000MI to 20000MI. In TimeOptimized, the user tries to minimize the average turnaround time of a job. In BudgetOptimized, user will try to minimize the average budget spent per job. Figure 6 shows the comparison of the job success rate in TimeOptimized and Random policies. Here success rate is defined as the number of jobs finishing within their deadline. Figure 7 shows the comparison of average turnaround time per job in TimeOptimized and Random policies. Figure 8 shows the comparison of the job success rate in BudgetOptimized policy. Figure 9 shows the comparison of budget spent per job in BudgetOptimized and Random policies.

M/c MIPS Rating	400	800	1200	1600	2000
Cost/Sec	2	6	10	14	18
No of machines	3	3	3	3	3

Table 1: MIPS and cost of each machine

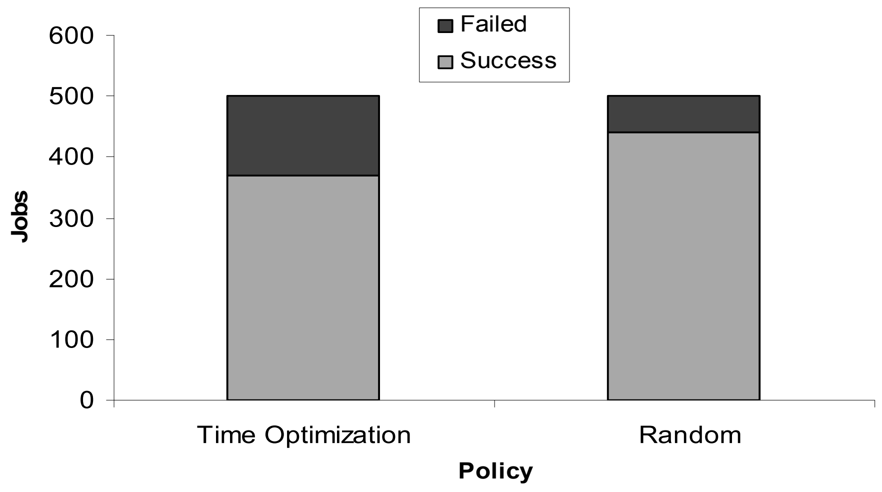

Figure 6: Job success rate in TimeOptimized and Random policies

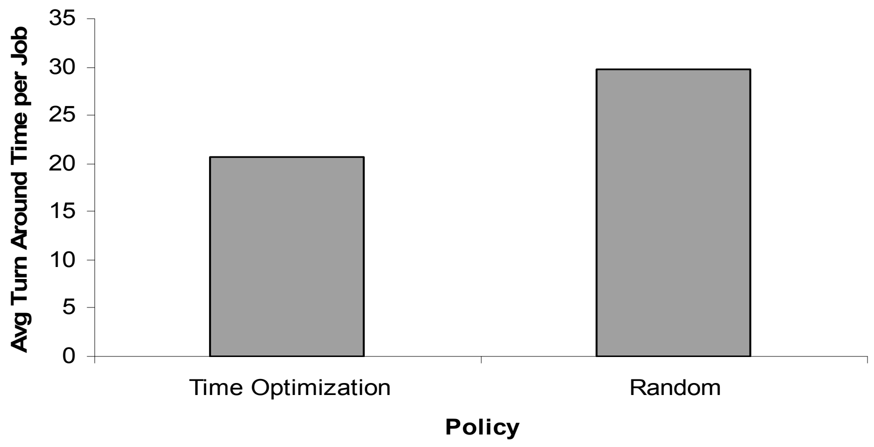

Figure 7: Average turnaround time per job in TimeOptimized and Random policies

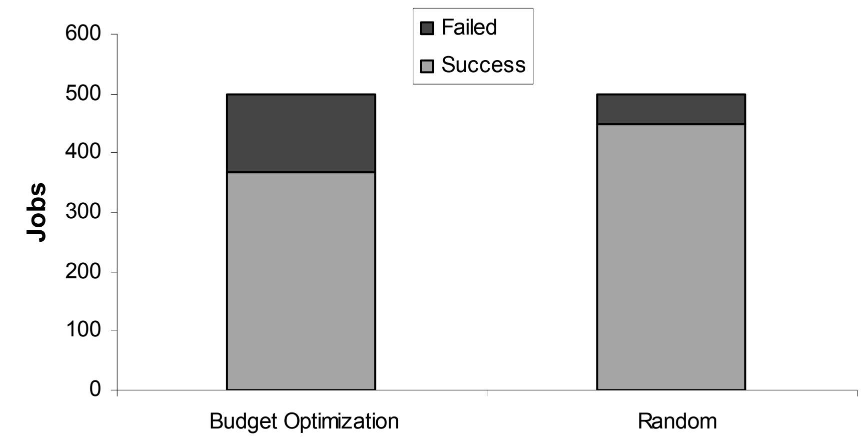

Figure 8: Job success rate in BudgetOptimized and Random policies

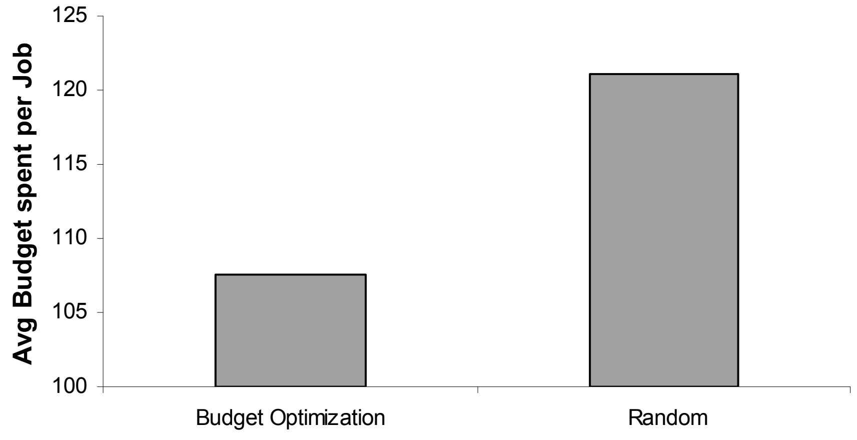

Figure 9: Average budget spent per job in BudgetOptimized and Random policies

From Figure 6 and Figure 7 it is clear that TimeOptimized policy reduces the average turnaround time but the number of jobs finishing within deadline is less than in Random policy. Similarly, in BudgetOptimized policy, the average budget spent per job is less than Random policy but the number of jobs finishing within deadline is also less in BudgetOptimized policy. The advantage of Random policy is that as each user selects the resource randomly, overall there will be less contention for resources, so most of the time the users will win in auctions. In TimeOptimized policy there will be high contention for high speed resources and only one user wins in auction and the rest who participated in the auction will fail. These failed jobs again participate in new auction. This process repeats and eventually they miss the deadline. A Similar phenomenon happens in BudgetOptimized policy. The justification for the above policies is explained below.

The probability of a user U_i winning in auction A_j is given by $(\ ) \sum_{k=1}^{u}$ [illegible]. Where B_i is the bid of U_i, u is total number of users participating in the auction and $P(A_j)$ is probability of choosing Auction *j*. In the above expression $\sum_{k=1}^{u}$ [illegible] refers to the probability of winning in the auction. If we consider that the budget allocations for users follow the uniform distribution then the probability of winning in A_j is proportional to the number of users participating in the grid. In both the policies, the probability of winning the resource is same but the probability of choosing the resource varies. In the TimeOptimized policy the probability P(A_j) is dependent on the capability of resource. Highly capable resources will have high probability of being chosen as they will complete the jobs earlier. In random policy the probability P(A_j) of choosing auction *j* is *1/n* (*n* is the total number of auctions). Similar explanation holds for BudgetOptimized policy.

Thus the number of jobs finishing within deadline in TimeOptimized and BudgetOptimized policies is less when compared with the Random policy. Random selection maximizes chances of winning but the average turnaround time will be higher than TimeOptimized policy and average budget per job (cost of acquiring the resources) will be higher than BudgetOptimzied policy.

In the next chapter, we propose two new policies. The first one, NewTimeOptimized policy, tries to minimize the average turnaround time while increasing the number of jobs finishing within their deadlines. The second one NewBudgetOptimized policy, tries to minimize the average budget spent per job while increasing the number of jobs finishing within their deadlines.

Chapter 5

NEWTIMEOPTIMIZED AND NEWBUDGETOPTIMIZED POLICIES

In this chapter we proposed two policies. The first one, NewTimeOptimized, tries to minimize the average turnaround time while increasing the number of jobs finishing within their deadlines. The second one NewBudgetOptimized, tries to minimize the average budget spent per job while increasing the number of jobs finishing within their deadlines.

5.1. NewTimeOptimized Policy

In NewTimeOptimized policy we sort the resources according to completion time and then we select one resource randomly from the top k resources for bidding. Here the user will specify the value of k. For $k = 1$ this policy is equivalent to the TimeOptimized case and for $k = n$ this policy is equal to the Random policy; where n is the total number of resources in the system. When we change k from 1 to n, effectively we are moving from TimeOptimized to Random policy. As k increases, the number of jobs finishing within their deadline increases, but the average turnaround time also increases. The pseudocode for this policy is shown in Figure 9.

5.1.1. Experiment Results

We conducted the experiment for three different job lengths, 10000MI to 20000MI, 50000MI to 100000MI and 100000MI to 200000MI. These three different job lengths reflect jobs with small, medium, and large CPU requirements. For each experiment, we varied k from 1 to 15 and we measured the average turnaround time and job success rate. Here success rate is defined as the number of jobs finishing within their deadline. It shows how the success rate and average turnaround time varies when we move from

TimeOptimized policy to Random policy. Figure 11 and Figure 12 show the job success rate and average turnaround time respectively for different values of *k* and for different job lengths.

```
S = { NULL };

// Find set of resources who can complete the job within deadline and budget
for (i=0; i<n; i++){

    // minimum time required to execute the job
    exec_time = job_length / R_i.speed;

    // completion time on Resource i
    completion_time = R_i.resource_usage_start_time + exec_time;
    if ((budget>=(R_i.price*exec_time)) AND (completion_time <=deadline) ){
        S = S ∪ R_i;
    }
}

sort S by completion_time;

// we will keep top k resources and remove rest of the resources
for (i=k+1; i<S.length; i++)
    remove S_i;
select randomly one resource R_i from S;
use complete amount allocated to job to bid R_i;
```

Figure 10: Algorithm for NewTimeOptimized policy

From Figure 11 and Figure 12 it is clear that when we go from k = 1 to k = 15 the success rate is increasing and at the same time the average turnaround time is also increasing. This is because, at k = 1, all users go for high capability resources and the competition for those resources will be high. Only one user in the auction will win and the rest of the resources who participated in the auction will lose, and to complete the job they have to again participate in another auction. This will repeat and eventually some jobs will miss the deadline. When we increase *k*, users will start choosing resources

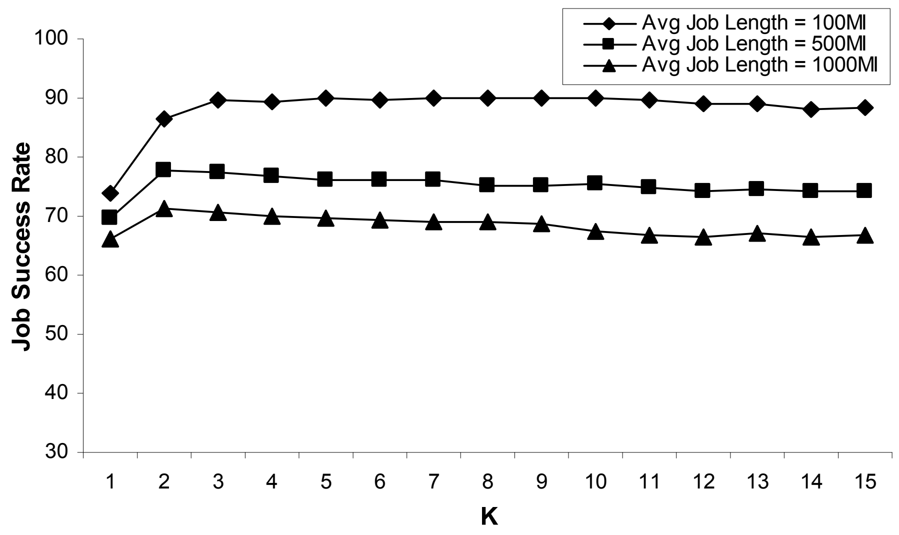

Figure 11: Job success rate in NewTimeOptimized policy for different values of *k*

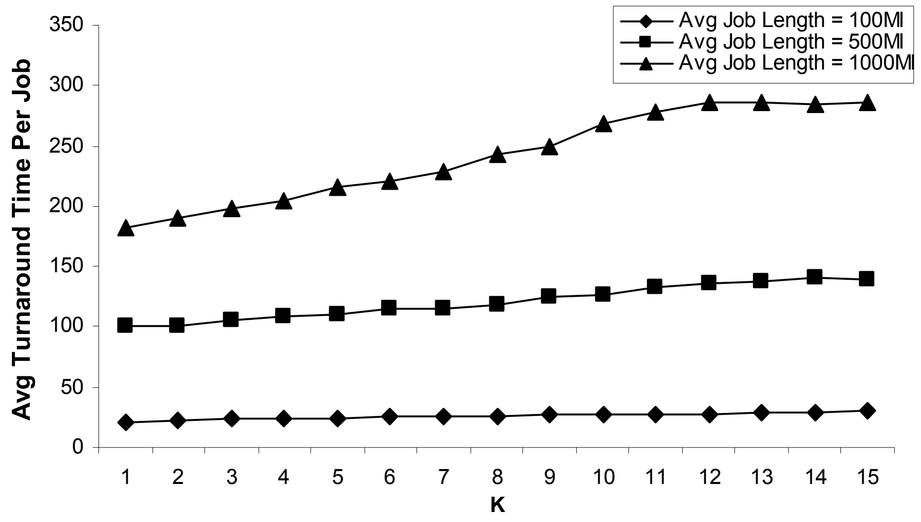

Figure 12: Average turnaround time in NewTimeOptimized policy for different values of *k*

randomly and the competition for resources will become less. At the same time the average turnaround time is increasing because of choosing lower capability resources.

At k = 4 the success rate is fairly high than at k = 15. At k = 15 all the users will choose resources randomly and distribution of jobs is random. Here distribution of jobs means, resources with high capability can execute more number of jobs than resources lower capability. Ideally any policy should distribute more number of jobs to high capable resources and less number of jobs to low capable resources. But when we follow Random policy, jobs will not get distributed in the above manner. So the number of jobs executing is higher at k = 4.

5.2. NewBudgetOptimized Policy

In NewBudgetOptimized policy, we sort resources according to cost and we select one resource randomly from the top k resources for bidding. At $k = 1$ this policy is equivalent to BudgetOptimized policy, and at $k = n$ this is equal to the Random policy. When we change k from 1 to n we are moving from BudgetOptimized to Random policy. As k increases, the number of jobs finishing within deadline increases but the average budget spent per job also increases. The pseudocode for NewBudgetOptimized policy is shown in Figure 13.

5.2.1. Experiment Results

Similar to NewTimeOptimized policy, we conducted the experiment for three different job lengths. For each experiment, we changed k from 1 to 15 and we measured out the average budget spent per job, job success rate, and the average turnaround time. Figure 14, Figure 15, and Figure 16 shows how the average budget spent per job, success rate, and the average turnaround time varied when we move from BudgetOptimized policy to Random policy.

```
S = { NULL };

// Find set of resources who can complete the job within deadline and budget
for ( i=0; i<n; i++){

    // minimum time required to execute the job
    exec_time = job_length / $R_i$.speed;

    // completion time on Resource i
    completion_time = $R_i$.resource_usage_start_time + exec_time;
    if((budget >=($R_i$.price * exec_time)) AND (completion_time<=deadline) ){
        $S = S \cup R_i$;
    }
}
sort S by price;
max_speed = $R_{S.length}$;

// we will keep top k resources and remove rest of the resources
for (i = k+1; i < S.length; i++)
    remove $S_i$;
select randomly one resource $R_i$ from S;
penalty = $R_i$.speed / max_speed;

// minimum amount needed to complete the job
min_amount_needed = (job_length/$R_1$.speed)*$R_1$.price;
bid_amount = budget – (budget –min_amount_needed)*penalty;
use bid_amount to bid for $R_i$;
```

Figure 13: Algorithm for the NewBudgetOptimized Policy

From Figure 14, Figure 15, and Figure 16 it is clear that when we go from *k* = 1 to *k* = 15, the budget spent per job will be increased and number of jobs getting executed will also be increased. Similar to TimeOptimized policy, at *k* = 4, the number of jobs is high and at the same time budget spent per job is less. The reason for this is the same as in NewTimeOptimized policy. As we increase *k*, the competition for low cost resources will be decreased, so the number of jobs getting executed will be increased. As we increase *k*, we start

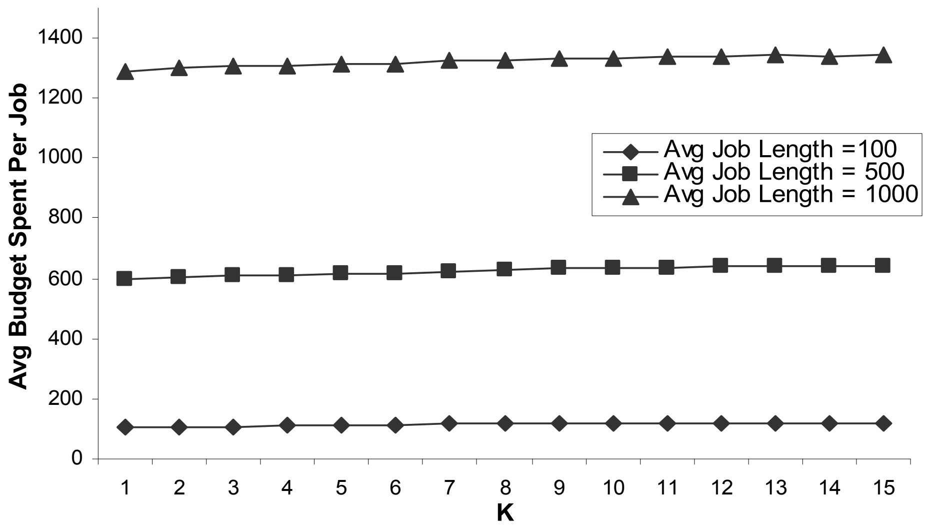

Figure 14: Average budget spent per job in NewBudgetOptimized policy for different values of *k*

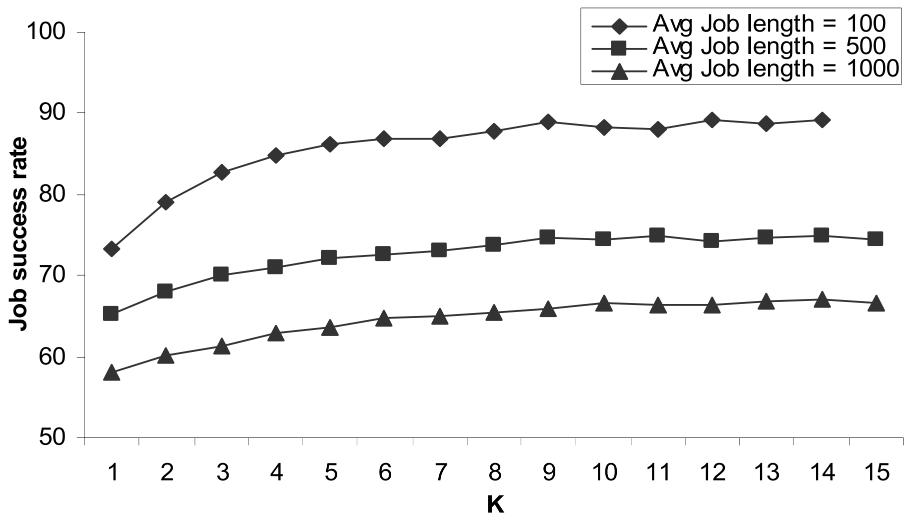

Figure 15: Job success rate in NewBudgetOptimized policy for different value of *k*

choosing resources randomly so the budget spent per job will be increased. The average turnaround time is very high at *k* = 1 because of choosing low cost resources. The cost of resources is proportional to resource speed. When we increase *k*, we choose resources randomly so the average turnaround time will be decreased.

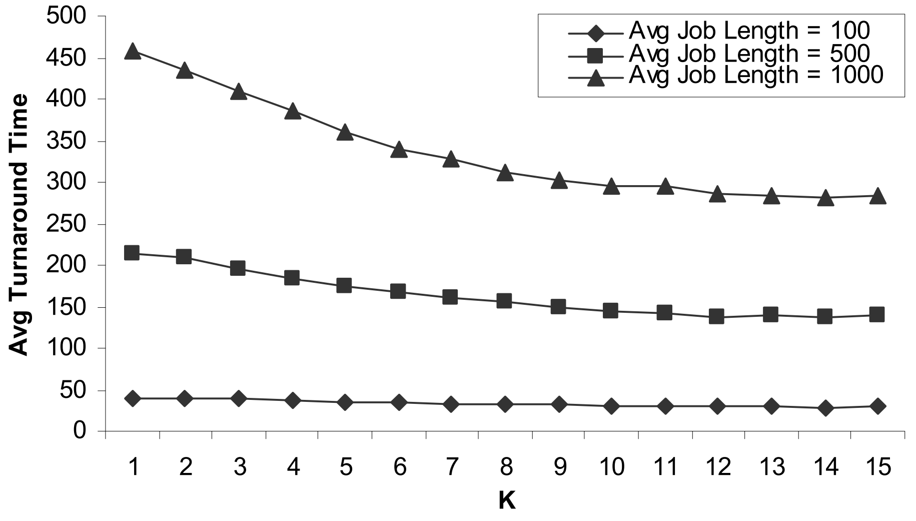

Figure 16: Average turnaround time in NewBudgetOptimized policy for different values of *k*

The above two policies either optimize the average turnaround time or average budget spent per job but not both. In the next chapter, we combined above two policies using AHP (Analytic Hierarchy Process) to take into account user preferences.

Chapter 6

RESOURCE SELECTION POLICY USING AHP

In the NewTimeOptimized and NewBudgetOptimized policies, we optimized only one parameter, either average turnaround time or average budget spent per job. We next implemented a resource selection policy that tries to decrease the average turnaround time and average budget spent per job based on user preference.

A user may not need to use just the TimeOptimized or BudgetOptimized policies alone. Usually a user wants to achieve a certain performance, but may also want to save on money spent at the same time. Thus, a user will usually have some relative preference for speed over time or vice versa. In our case, the user will specify the preference for time vs. budget. We have used Analytic Hierarchy Process (*AHP*) [9, 21] technique to model the user preference.

6.1. Analytic Hierarchy Process

AHP, developed by Saaty [14], is a mathematical technique for multicriteria decision making. The structure of AHP consists of a hierarchy of criteria and sub-criteria cascading from the decision objective or goal. By making pairwise comparisons at each level of the hierarchy, we develop relative weights, called priorities, to differentiate between the importance of the criteria. Here criteria are nothing but the parameters on which we want to make our decision. In our case, we are considering two parameters, time and budget, and the user will give preference to time vs. budget. As shown in Table 2, we form a pairwise comparison of each criterion, where the i^{th} row and the j^{th} column give the relative weight of criteria's C_i and C_j. The weights are assigned on the relative scale between 1 and 9; 1 means equal importance and 9 means extreme important. Full details of the AHP can be found in [15]. The advantage of the AHP is we can combine the qualitative and quantitative information. Although in our paper we are considering quantitative information like cost and speed of resources we can also use qualitative information like reliability of the resource etc. In the rest of this chapter, we used the term *criteria* to describe the parameters on which we want to take decision and *alternatives* refer to the resources which we want to choose.

	C_1	C_2	….	C_n
C_1	1	C_{12}	….	C_{1n}
C_2	$1/C_{12}$	1	….	C_{2n}
:	:	:	:	:
C_n	$1/C_{1n}$	$1/C_{2n}$	….	1

Table 2: Example of pairwise comparison matrix

	Budget	**Time**	**Normalized Values**
Budget	1	1/9	0.16102
Time	9	1	0.83898

Table 3: Preference of time and budget

6.2. Steps in AHP

In this section, we have explained the AHP by taking an example. Let us assume that there are 3 resources and we have to choose one resource among them for bidding. Let *R1*, *R2*, *R3* be the available resources and their cost, speed and their assigned weights are shown in Table 4. The problem is that we have to choose one resource for bidding according to user preference.

1. The first step in AHP is to decide the criteria or parameters by which one is chosen among the alternatives. In this problem, one resource should be

selected for bidding based on the average turnaround time and cost of the resources.

2. The second step is to determine the relative weights of each of the criteria compared. The relative weights are taken from the scale 1 to 9. 1 means equal preference and 9 means extremely high preference. The AHP uses pairwise comparison technique. If user has given time to budget preference as 9 means, he has given high preference to turnaround time. The pairwise comparison matrix for our problem is shown in Table 3.
3. The third step is to compare the alternatives based on the each selected criteria. In our case for each criterion we will compare the resources based on the average turnaround time and cost.

Resource	R1	R2	R3
Budget (rank)	2 (3)	6 (2)	10 (1)
Time (rank)	400 (1)	800 (2)	1200 (3)
Budget weight log(2*WeightRank)	0.77815	0.602059	0.301029
Speed Weight log(2*SpeedRank)	0.301029	0.602059	0.778155

Table 4: Weights assigned to resources

In this problem, the resources are not compared based on the absolute scale. For example in the case of budget, the resources are not directly compared based on their cost of the resources. First the resources are ranked according to the speed and cost of the resources. The highest speed or the lowest cost resource will get a high rank. Resources with equal capability will get equal rank. The reason for doing this is as follows. Suppose that we have resources with speed 400MIPS, 900MIPS, and 3000MIPS. If the resources are compared based on the absolute scale, 3000MIPS resource will get very high preference. To avoid this, we are considering the resource ranks instead of their absolute values. After this we have taken the logarithmic value of the rank. The advantage of doing this is that in the logarithmic scale, the slope of the curve will decrease when we increase the values. In other words, the resources with high capability will get high preference and it will become less and less as we go down towards lower capability resources. Table 4 shows the rank and weight of the resources. After assigning the weights we have to compare the resources pairwise based on these weights. By comparing the resources pairwise, we will get the relative importance of one resource over other. These relative values should be normalized before using. Table 5 and Table 6 show the pairwise comparison of resources based on the speed and budget respectively. Let us assume Ws_i is the relative weight of alternative i with respect to speed and Wb_i is weight of alternative i with respect to budget. Mathematically we can express the above as:

$$Ws = Ws_1 + Ws_2 + \ldots + Ws_n;$$

$$Wb = Wb_1 + Wb_2 + \ldots + Wb_n;$$

where

$$Ws_i = 1 \Big/ \sum_{j=1}^{n} \frac{\log(Rs_i)}{\log(Rs_j)}$$

with Rs_i as the rank of alternative i with respect to speed, and

$$Wb_i = 1 \Big/ \sum_{j=1}^{n} \frac{\log(Rb_i)}{\log(Rb_j)}$$

where Rb_i is the rank of alternative i with respect to budget.

	R1	R2	R3	Normalized Values
R1	1	1.292481	2.584967	0.462843
R2	0.773706	1	2.000003	0.358105
R3	0.386852	0.499999	1	0.179052

Table 5: Pairwise comparison of resources based on speed of the resources

	R1	R2	R3	Normalized Values
R1	1	0.499999	0.38685	0.179051
R2	2.000003	1	0.773701	0.358103
R3	2.584984	1.29249	1	0.462845

Table 6: Pairwise comparison of resources based on budget of the resources

4. The final step is to take the weighted average of the relative weights obtained in step 2 with the above normalized values. Table 7 shows the above process for the example we have taken. The resource with the highest weight is chosen for bidding. Based on the given time to budget preference, *R3* will get the highest weight, meaning that *R3* is the most suitable resource. Instead, if the time to budget preference is 1, then *R2* will get the highest weight. The pseudocode for choosing the resources based on the AHP is explained in Figure 17. In the pseudocode, R_i.*speed_rank* refers to rank of resource *i* based on speed, R_i.*cost_rank* refers to rank of resource *i* based on cost, R_i.*speed_weight* refers to weight assigned to resource *i* based on speed, and R_i.*cost_weight* refers to weight assigned to resource *i* based on cost. Mathematically we can express the above as:

$$Pa_i = \left(Ws_i \times UPtb\right) + \left(Wb_i \times (1\text{-}UPtb)\right)$$

where Pa_i is the final weights of alternative *i* and *UPtb* is the time to budget preference.

	Cost Values	Speed Values		Attribute Weights		Final Weights
R1	0.462	0.179		0.161	=	0.225
R2	0.358	0.358	X			0.358
R3	0.179	0.463		0.839		0.417

Table 7: Weighted average of cost, speed values with attribute weights

```
Let $S$ be the set of resources that user can complete the job within deadline and budget;
Let assume that $A$ be the normalized attribute matrix;
For each $R_i$ in $S$, find its rank $R_i$.speed_rank and $R_i$.cost_rank according to speed and cost;
for (i=0; i<S.length; i++){
    $R_i$.speed_weight = log(2* $R_i$.speed_rank);
    $R_i$.cost_weight = log(2* $R_i$.cost_rank);
    total_speed_val += $R_i$.speed_weight;
    total_cost_val += $R_i$.cost_weight;
}
V = {0};
for (i=0; i<S.length; i++){
    V[i][0] = $R_i$.cost_weight / total_cost_val;
    V[i][1] = $R_i$.speed_weight / total_speed_val;
}

// Multiply the above two matrices and W gives the final weights of alternatives
W = V x A;
select $R_i$ from S which has maximum weight;

//max_speed is the maximum speed of the resource that the user can bid
penalty = $R_i$.speed / max_speed;

// minimum amount needed to complete the job
min_amount_needed = (job_length/ $R_i$.speed)* $R_i$.price;
bid_amount = budget – (budget –min_amount_needed)*penalty;
use bid_amount to bid for $R_i$;
```

Figure 17: Algorithm for the combined policy

6.3. Experimental Results

First, the time to budget preference is varied from 9 to 1/9 for all users. The experiment is conducted for three different job lengths; 10000MI to 20000MI, 50000MI to 100000MI and 100000MI to 200000MI. Time to budget preference of 9 means there is high preference for the turnaround time of the resources and time to budget preference of 1 means, there is equal preference for turnaround time and cost of the resources. When we change the value from 9 to 1/9, effectively we are moving from TimeOptimized to BudgetOptimized policy. Figure 18 shows the job success rate with respect to time to budget preference. Figure 19 shows the average turnaround time of jobs for different values of time to budget preference. Figure 20 shows the average budget per job for different values of time to budget preference.

From Figure 19 it is clear that when the time vs. budget preference is changing from TimeOptimized to BudgetOptimized policy, the average turnaround time per job is increasing. This is because as the value is changing from TimeOptimized to BudgetOptimized policy, we are choosing resources with lower and lower speeds and the execution time on these low speed resources will be high which in turn increases the total turnaround time for the jobs. Figure 20 shows that the average budget spent per job is decreasing when the value is changing from from TimeOptimized to BudgetOptimized policy. This is because as the value changes from TimeOptimized to BudgetOptimized policy, we are choosing low speed resources and the price per MI will be lower and budget per job will be decreased. Effectively the resources are chosen according to user choice.

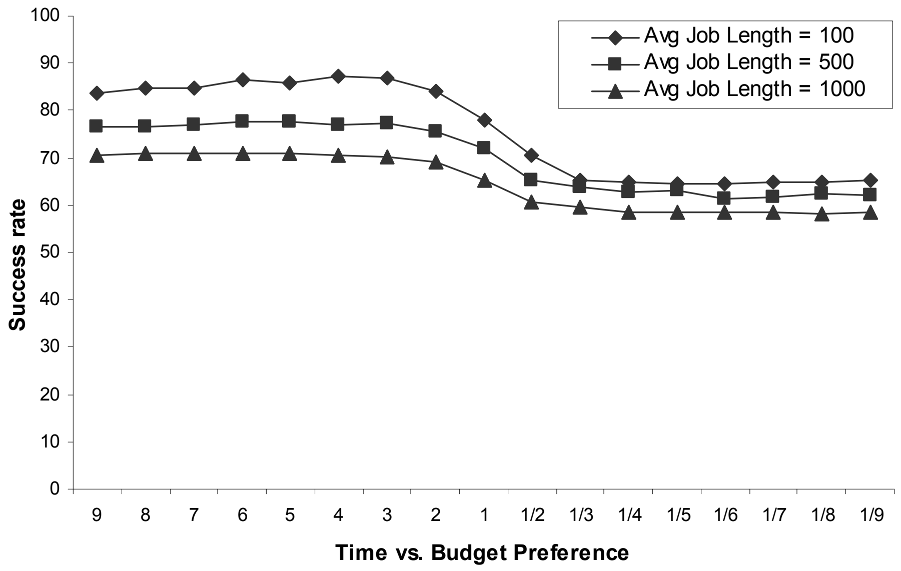

Figure 18: Job success rate for different values of time vs. budget preferences

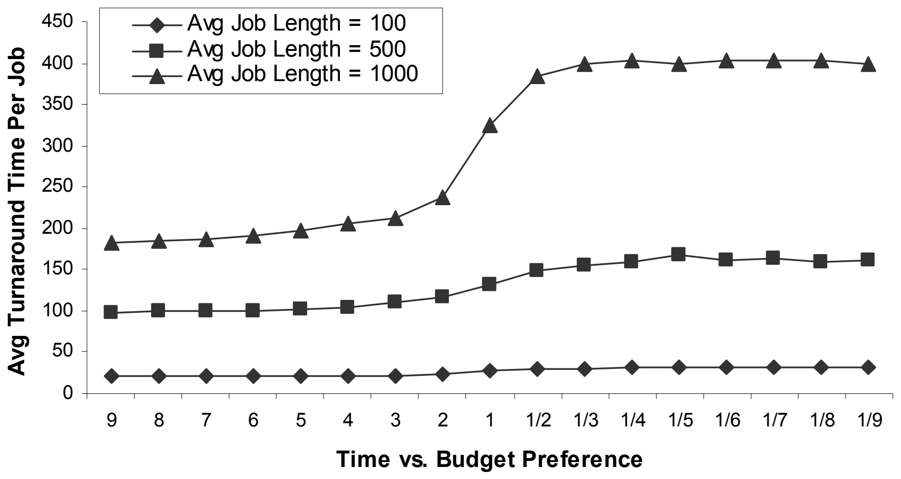

Figure 19: Average turnaround time for different values of time vs. budget preference

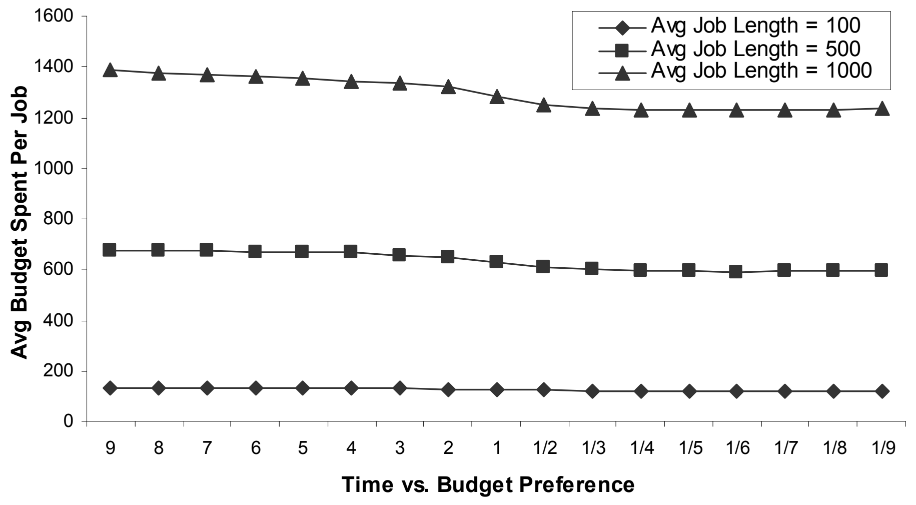

Figure 20: Average budget spent per job for different values of time vs. budget preference

From Figure 18, the job success rate is decreasing when the preference is changing from TimeOptimized to BudgetOptimized policy. This is because the execution time on a high speed resource is low so it can execute more number of jobs when compared with a low speed resource. In other words, the number of auctions conducted by a high speed resource in some time interval is more when compared with a low speed resource. So the success rate is decreasing when the preference is changing from TimeOptimized to BudgetOptimized policy. Let us assume that three jobs J_1, J_2, and J_3 are participating in an auction and assume that their job length is 150MI and its deadline 100 sec. First, assume that all the jobs are participating in the auction conducted by the 2000MIPS resource and its default auction time is 30 sec. Let assume that *J1* first won in the auction and then *J2* and *J3*. In that case *J1* will complete the job by 37.5 sec, *J2* will complete the job by 67.5 sec and *J3* will complete the job by 97.5 sec. Instead if all participated in the auction conducted by 400MIPS resource, in that case only *J1* and *J2* can complete the job. *J3* will miss the deadline. This is because the execution time on the 400MIPS resource will be more and all the jobs will have to wait more time to get the resource.

If TimeOptimized policy is compared with combined policy at the time to budget preference of 9, except the job success rate, there is not much difference in the average turnaround time and average budget spent per job. But the job success rate is increased around 83% in the combined policy as compared to around 74% in TimeOptimized policy. The difference between these two can be explained by considering the demand for the resources. From Table 7, the number of auctions with single user participation is almost 10% higher than TimeOptimized policy and this is the main reason for increase in the success rate in the combined policy. The reason is, although time to budget preference of 9 means high importance for turnaround time, still there is a small percentage of preference we have given to budget and this is the reason for the difference in the success rate.

No of Participants	Frequency	Cumulative %
1	281	71.50%
2	81	92.11%
3	19	96.95%
4	10	99.49%
5	2	100.00%
6	0	100.00%

Table 8: Percentage of users participating in the auctions in the combined policy

No of Participants	Frequency	Cumulative %
1	217	60.78%
2	84	84.31%
3	30	92.72%
4	20	98.32%
5	5	99.72%
6	1	100.00%

Table 9: Percentage of users participating in the auctions in TimeOptimized policy

At higher job lengths, the ratio of the number of jobs succeeded to the number of jobs failed at time vs. budget preference of 9 to 1/9 is reduced when compared with lower job lengths. This is because, when the job size is increased, the advantage of high speed resource as explained in the above will be reduced if the job size is high.

In the second experiment, we have given different time to budget preferences for users as shown in Table 9. The job length for this experiment is varied from 10000MI to 20000MI. This experiment is performed to show that the user is getting the resource it wants according to its preference. From Figure 21, the average budget spent per job is decreasing when the user chooses low speed to budget preference. From Figure 22, the average turnaround time per job is increasing when he chooses low time to budget preference. Figure 23 shows the success rate for users. Unlike our previous experiment, the success rate is not decreasing when we decrease the time vs. budget preference. The reason for this is, unlike our previous experiment, all the users are not having the same preference and there will be less contention for resources.

We can improve the success rate in the above experiments by using the strategy employed in the NewTimeOptimized and NewBudgetOptimized policies i.e. we can select one resource from the top *k* resources randomly.

User	Time vs. Budget Preference
User 1	9
User 2	7
User 3	5
User 4	3
User 5	1
User 6	1/3
User 7	1/5
User 8	1/7
User 9	1/9
User 10	1/10

Table 10: Time vs. Budget preference for users

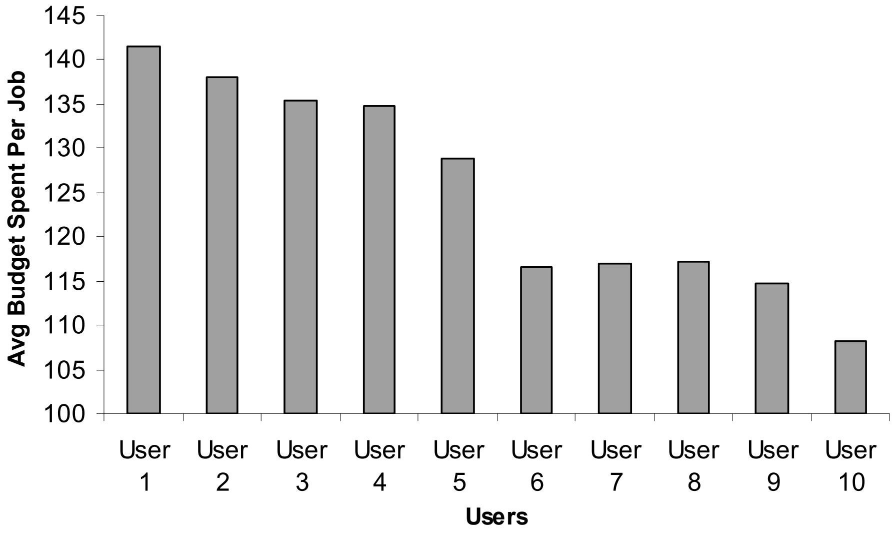

Figure 21: Average budget spent per job for users with different time vs. budget preference

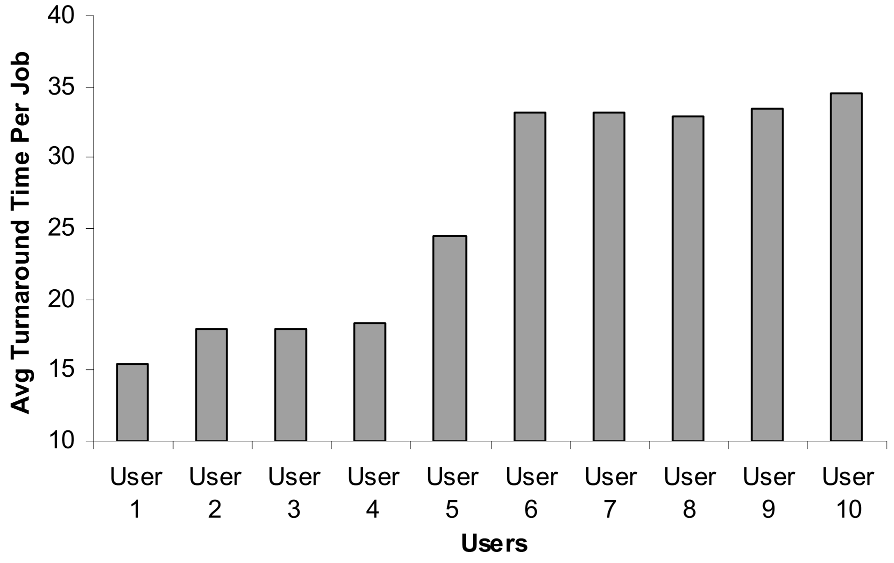

Figure 22: Average turnaround time per job for users with different time vs. budget preference

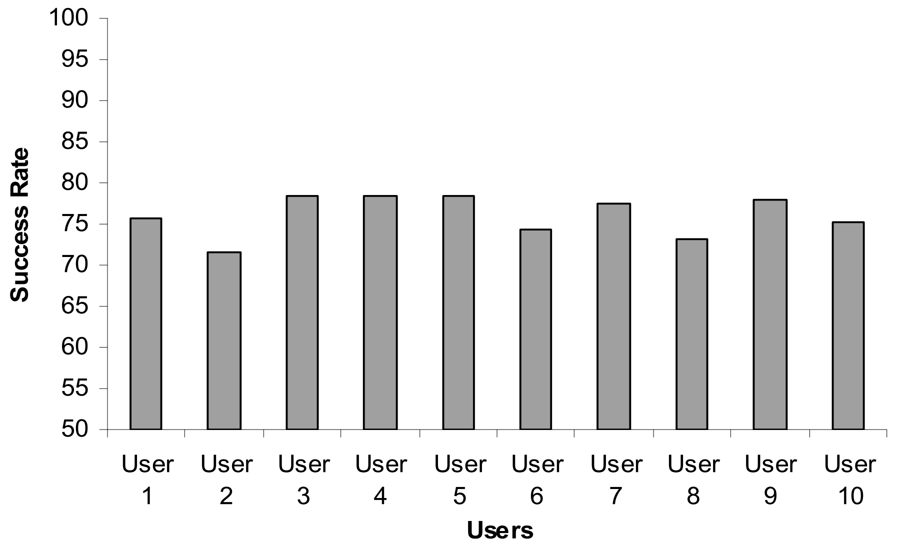

Figure 23: Success rate for users with different time vs. budget preference

6.4. Extending to More Than Two Parameters

In our present work, the decision is taken by considering only two parameters, speed and budget of the resources. It can be extended similarly for other parameters also. For example we can include available memory in the above case. The same methodology that is used in the above case can be used for this also i.e. sort the resources according to available memory and assign weights to it as explained in the step3 of section 6.2. The next step is to add one more column and row in the attribute matrix. We have to compare the speed with memory and budget with memory and finally we will take the weighted average of attribute matrix with the individual values formed based on speed, cost, and memory of the resources. In a similar way we can include other parameters also. In general,

$$[illegible] \sum_{i=1}^{m}\sum_{j=1}^{n} [illegible]$$

where P_i is the preference of criteria i, W_{ij} is the relative weight of alternative i with respect to criteria j.

AHP can be used to combine qualitative and quantitative information easily. For example assume that we want to include qualitative parameters like reliability of resources. The same procedure as explained above for qualitative parameters can be used for reliability of resource with one exception. The alternatives can not be compared based on qualitative parameter, so we have to express it in terms of quantitative information. The relative preference of one alternative over other can be expressed in a scale from 1 to 9.

In the next chapter, we presented one history based technique that considers the previous success rate for choosing resources.

Chapter 7

HISTORY BASED POLICY

In this chapter we presented one history based technique that considers the previous success rate for choosing resources. Here the user will give preference in terms of the minimum percentage of jobs that should be finished within deadline.

7.1. History Based Policy

The main motivation for this policy is, the success rate in NewTimeOptimized policy is higher than NewBudgetOptimized policy but in NewTimeOptimized policy the user is not able to save money. In this policy after getting the minimum success rate we are trying to save money by shifting to NewBudgetOptimized policy.

Initially we start with NewTimeOptimized policy with $k = 4$ and when the success rate exceeds the user specified value we shift to NewBudgetOptimized policy with $k = 4$. In NewBudgetOptimized we do not use our complete budget for a job to bid. So first we are using our complete amount for bidding in the initial stage, and after getting sufficient success rate we are shifting to NewBudgetOptimized policy. This policy effectively tries to reduce the average budget spent per job while maintaining the user requirements. If the user specifies high success rate then this policy is then same as NewTimeOptimized policy and if user specifies very low success rate then this policy is same as NewBudgetOptimized policy.

The pseudocode for this policy is shown in Figure 24. In the algorithm, the penalty will be very less for NewTimeOptimized policy and we almost use our complete amount allocated to the job for bidding. But if we use

NewBudgetOptimized policy then the penalty will be higher and it will not use the complete amount allocated for bidding.

```
$S$ = { NULL };

// Find set of resources who can complete the job within deadline and budget
for ($i$=0; $i<n$; $i$++){

    //  minimum time required to execute the job
    exec_time = job_length / $R_i$.speed;

    // completion time on Resource $i$
    completion_time = $R_i$.resource_usage_start_time + exec_time;
    if((budget >=($R_i$.price * exec_time)) AND (completion_time<=deadline) ){
        $S = S \cup R_i$;
    }
}

success_rate = no_jobs_finished / total_no_jobs;
if(success_rate <= user_success_rate)
    Sort $S$ by completion_time;
else
    Sort $S$ by budget;

// we will keep top k resources and remove rest of the resources
for ($i = k+1$; $i < S$.length; $i$++)
    remove $S_i$;
select randomly one resource $R_i$ from $S$;

// penalty will be very less if we choose TimeOptimized and we use almost all the amount for bidding
penalty = $R_i$.speed / max_speed;

// minimum amount needed to complete the job
min_amount_needed = (job_length/$R_1$.speed)*$R_1$.price;
bid_amount = budget – (budget –min_amount_needed)*penalty;
Use bid_amount to bid for $R_i$;
```

Figure 24: Algorithm for history based policy

7.2. Experimental Results

We compared the proposed adaptive policy with NewTimeOptimized policy. In adaptive policy, we have taken the user specified success rate as 70%, meaning that the resource broker will initially try to get 70% success rate by using NewTimeOptimized policy and after getting that, it will shift to NewBudgetOptimized policy and tries to decrease the total amount spent. Figure 25 compares the adaptive policy with NewTimeOptimized on the success rate, turnaround time, and average budget per job for average job length of 10000MI. Similarly Figure 26 and Figure 27 show the same for average job length of 50000MI and 100000MI respectively.

From Figure 25, for user specified success rate of 70%, the actual success rate obtained is around 84%, which is slightly less when compared to NewTimeOptimized policy but the average budget per job is decreased by around 12% when compared to NewTimeOptimized policy. This is because, after getting sufficient success we are switching to NewBudgetOptimized policy and we are trying to save money after that. With this policy we are missing some of jobs but at the same time we are reducing the average budget spent per job.

The same explanation is true for budget 50000MI and for 100000MI job lengths. But for 100000MI job lengths, the actual success rate is less than the user specified success rate but it is very close to it. This is because the success rate for NewTimeOptimized policy is exactly 70% and in our history based policy the success rate is usually less than NewTimeOptimized. So in this policy it is slightly reduced.

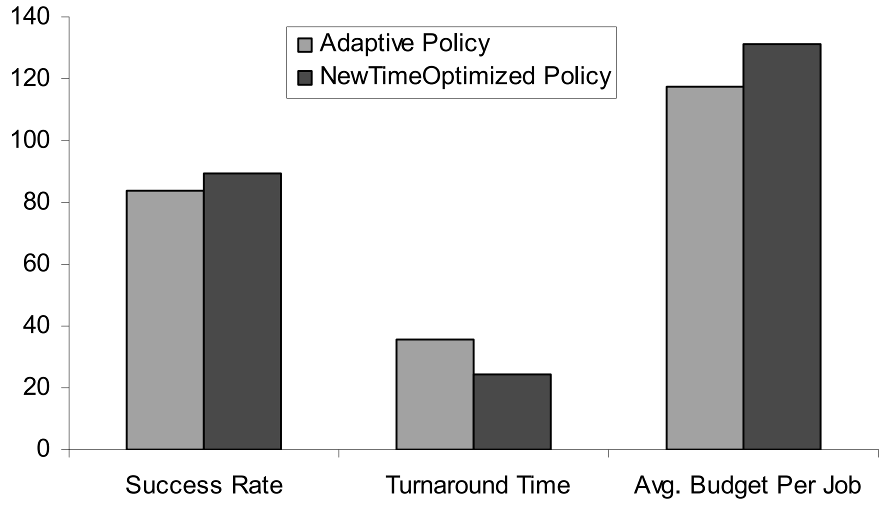

Figure 25: Comparison of NewTimeOptimized policy with Adaptive policy for average job length of 10000MI

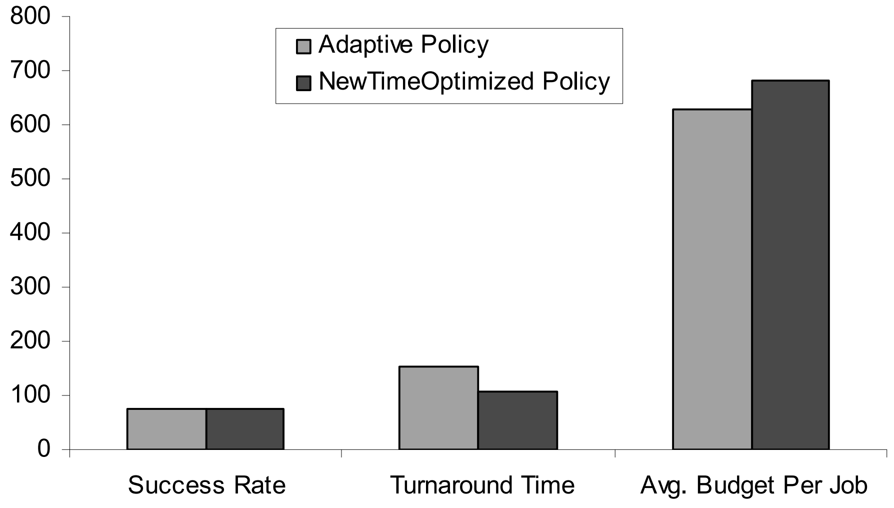

Figure 26: Comparison of NewTimeOptimized policy with Adaptive policy for average job length of 50000MI

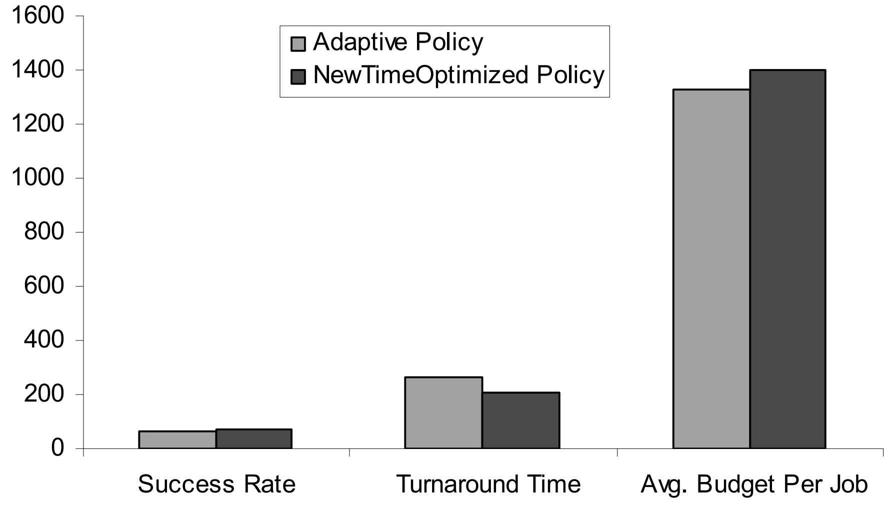

Figure 27: Comparison of NewTimeOptimized policy with Adaptive policy for average job length of 100000MI

Chapter 8

CONCLUSION & FUTURE WORK

In this thesis, we proposed and evaluated several auction based resource selection policies for users in grid. In all these policies we tried to optimize the average turnaround time, the average budget spent per job, and the success rate. First we proposed simple TimeOptimized, BudgetOptimized, and Random policies. The success rate of these policies is less than Random policy but the average turnaround time and average budget spent per job is more in Random policy. We next proposed NewTimeOptimized and NewBudgetOptimized policies that tries to increase the number of jobs getting successful and at the same time it reduces the average turnaround time and average budget spent per job respectively. Next we proposed a policy that tries to optimize the average turnaround time and average budget spent per job simultaneously based on the user preference. Finally we proposed a history based policy that considers the previous success rate and depending on that it chooses either NewTimeOptimized or NewBudgetOptimized policy.

In our work we considered only two parameters, average turnaround time and average budget spent per job. However, our policies can be easily extended to include more parameters. In chapter 6, we explained this by taking an example on how to include more parameters into it. We evaluated each of the above policies through simulation and presented the results.

In future, we would like to improve the history based policy. At present the history based policy is not considering the previous bid information for future bidding. We can find the relative demand of the resources by considering the previous bid. In this way we can behave more strategically while bidding. We would like to investigate the scalability of auction for larger number of users

& resources and to explore the use of other market mechanisms for resource allocation in grids.

REFERENCES

1. Amir Y., Awerbuch B., and Borgstrom R. S., "A Cost-Benefit Framework for Online Management of a Metacomputing System", *International Conference on Information and Computational Economy*, Oct. 1998.

2. Amir Y., Awerbuch B., Barak A., Borgstrom R.S., and Keren A., "An opportunity cost approach for job assignment in a scalable computing cluster", *IEEE Transactions on Parallel and Distributed Systems*, Jul. 2000, vol. 11, no. 7, pp. 760--??.

3. 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.

4. Berman F., Hey A. J. G., and Fox G., "Grid Computing: Making the Global Infrastructure a Reality", *John Wiley & Sons*, 2003.

5. Buyya R., Abramson D., Giddy J., and Stockinger H., "Economic Models for Resource Management and Scheduling in Grid Computing", *Concurrency and Computation: Practice and Experience (CCPE)*, Nov 2002, vol. 14, no. 13, pp. 1507-1542.

6. Buyya R., Abramson D., and Giddy J., "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", *International Conference on High Performance Computing in Asia-Pacific Region*, May 2000.

7. Buyya R. and Murshed M., "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", *Concurrency and Computation: Practice and Experience*, Nov. 2002, vol. 14, no. 13, pp. 1175-1220.

8. Chee S.Y., and Rajkumar Buyya R., "A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing", *Software: Practice and Experience*, 2005, Wiley Press, USA..

9. Chen M., Yang G., and Liu X., "Gridmarket: A Practical, Efficient Market Balancing Resource for Grid and P2P Computing", *International Workshop*

on Grid and Cooperative Computing (GCC 2003), Dec. 2003, Lecture Notes in Computer Science, vol. 3033, pp. 612–619.

10. Chen C., Maheswaran M., and Toulouse M., "Supporting co-allocation in an auctioning-based resource allocator for grid systems", *International Parallel and Distributed Processing Symposium*, 2002.

11. Clarke D. and Tangney B., "Microeconomic theory applied to distributed systems", *Technical Report TCD--CS--93--30*, 1993, Distributed Systems Group, University of Dublin.

12. Chunlin L. and Layuan L., "A Utility-Based Two Level Market Solution for Optimal Resource Allocation in Computational Grid", *International Conference on Parallel Processing (ICPP'05)*, 2005, pp. 23-30.

13. Das A. and Grosu D., "Combinatorial Auction-Based Protocols for Resource Allocation in Grids", *IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

14. Dumas M., Aldred L., Governatori G., Hofstede A., and Russell N., "A probabilistic approach to automated bidding in alternative auctions", *International Conference on World Wide Web*, 2002, pp. 99--108.

15. Ernemann C., Hamscher V., and Yahyapour R., "Economic Scheduling in Grid Computing". *8th International Workshop on Job Scheduling Strategies For Parallel Processing*, 2002, Lecture Notes In Computer Science, vol. 2537, pp. 128-152.

16. Ferguson D.F., "The Application of Microeconomics to the Design of Resource Allocation and Control Algorithms in Distributed Systems", *PhD thesis*, Columbia University, 1989.

17. Ferguson D.F., Nikolaou C., Sairamesh J., and Yemini Y., "Economic Models for Allocating Resources in Computer Systems", *Market Based Control of Distributed Systems*, World Scientific Press, 1996.

18. Figueira J., Greco S., and Ehrgott M., "Multiple Criteria Decision Analysis - State of the Art Surveys", *Springer International Series in Operations Research and Management Science,* vol. 76.

19. Foster I., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International Journal of High Performance Computing Applications*, 2002, vol. 15, no. 3.

20. Foster I., "What Is the Grid? A Three Point Checklist," *GridToday,* Jul. 2002, vol. 1, no. 6.

21. Foster I. and Kesselman C., "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 2004.

22. Gibbins H., Nadiminti K., Beeson B., Chhabra R., Smith B., and Rajkumar Buyya, "The Australian BioGrid Portal: Empowering the Molecular Docking Research Community", *APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch (APAC 2005)*, Sept. 2005.

23. Gomoluch J., and Schroeder M., "Market-Based Resource Allocation for Grid Computing: A Model and Simulation", *International Workshop on Middleware for Grid Computing (MGC `03),* 2003.

24. Greenwald A. and Stone P., "Autonomous bidding agents in the Trading Agent Competition", *IEEE Internet Computing*, Apr. 2001.

25. Grosu D. and Das A., "Auction-Based Resource Allocation Protocols in Grids," *International Conference on Parallel and Distributed Computing and Systems*, Nov. 2004, pp. 20–27.

26. Heiser G., Lam F., and Russell S., "Resource management in the mungi single-address-space operating system", *Australian Computer Science Conference*, Feb. 1998, pp. 417-428.

27. Huberman B.A, Hogg T., and Swami A., "Using Unsuccessful Auction Bids to Identify Latent Demand", *IEEE Conference on Systems, Man and Cybernetics*, 2001, pp. 2911-2916.

28. Kale L. V., Kumar S., Potnuru M., DeSouza J., and Bandhakavi S., "Faucets: Efficient Resource Allocation on the Computational Grid", *International Conference on Parallel Processing (ICPP 2004)",* Aug. 2004, pp. 396–405.

29. Kant U., Grosu D., "Double Auction Protocols for Resource Allocation in Grids", *International Conference on Information Technology: Coding and Computing (ITCC'05),* 2005, vol. 1, pp. 366-371.

30. Klemperer P., “Auction Theory: A Guide to the Literature”, *Journal of Economic Surveys*, July 1999, vol. 13, no. 3, pp 227 -- 286.

31. Krauter K., R. Buyya R., and Maheswaran M., “A Taxonomy and Survey of Grid Resource Management Systems”, *International Journal of Software: Practice and Experience*, 2002, vol. 32, pp. 135—164.

32. Lai K., Huberman B. A., and Fine L., “Tycoon: A Distributed Market-based Resource Allocation System”, Apr. 2004, *Technical Report*, HP Labs, USA.

33. Mankiw N.G., "Principles of Economics", The Dryden Press, 2003.

34. Nakai J., "Pricing Computing Resources: Reading between the Lines and Beyond," *Technical Report* NAS-01-010, *NASA Ames Research Center, Advanced Supercomputing Division,* Nov 2001.

35. Nisan N., “Algorithms for Selfish Agents”, *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, 1999, pp.1 -- 15.

36. Nisan N., London S., Regev O., and Camiel N. “Globally distributed computation over the internet -- the POPCORN project”, *International Conference on Distributed Computing Systems (ICDCS98)*, 1998.

37. Nisan N., London S., Regev O., and Camiel N., “Globally distributed computation over the Internet — the POPCORN project”, *International Conference on Distributed Computing Systems*, 1998.

38. Quétier B. and Cappello F., "A survey of Grid research tools: simulators, emulators and real life platforms", *IMACS World Congress (IMACS)*, 2005.

39. Reeves D.M., Wellman M.P., MacKie-Mason J.K., and Osepayshvili A., “Exploring bidding strategies for market-based scheduling”, *Decision Support Systems*, 2004.

40. Saaty T. L., “The Analytic Hierarchy Process”, McGrawHill, New York, 1980.

41. Sandholm T., “Algorithm for optimal winner determination in combinatorial auctions”, *Artificial Intelligence*, Jan. 2002, pp. 135:1—54.

42. Sandholm T., “Making markets and democracy work: A story of incentives and computing”, *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pages 1649—1671.

43. Schriber T. and Brunner D., "Inside DiscreteEvent Simulation Software: How It Works and Why It Matters", *Simulation Conference*, 1998, pp. 77--86.

44. Varian H., “Economic mechanism design for computerized agents”, *First USENIX Workshop on Electronic Commerce*, July 1995.

45. Waldspurger C.A., Hogg T., Huberman B.A., Kephart J.O., and Stornetta W.S., "Spawn: A Distributed Computational Economy," *IEEE Transactions on Software Engineering*, Feb. 1992, vol. 18, no. 2, pp. 103-117.

46. Wellman M.P., "Market-oriented programming: some early lessons", Market-based control: A paradigm for distributed resource allocation, World Scientific Publishers, 1996.

47. Wellman M.P., Walsh W.E., Wurman P.R., and MacKie-Mason J.K., "Auction protocols for decentralized scheduling", *Games and Economic Behavior*, 2001, vol. 35, pp. 271--303.

48. Wolski R., Plank J. S., Brevik J., and Bryan T., “Analyzing Market-Based Resource Allocation Strategies for the Computational Grid”. *International Journal of High Performance Computing Applications*, Aug. 2001, vol. 15, no. 3, pp. 258-281.

49. Xiao L., Zhu Y., Lionel M., and Xu Z., "GridIS: An Incentive-Based Grid Scheduling", *IEEE International Parallel and Distributed Processing Symposium (IPDPS'05),* 2005.

50. Yeo C.H. and Buyya R., "Pricing for Utility-driven Resource Management and Allocation in Clusters", *International Conference on Advanced Computing and Communication (ADCOM)*, 2004.

51. Deardorff's Glossary of International Economics, May 2006, http://www.personal.umich.edu/~alandear/glossary/m.html.