Cost Based Optimization of job allocation in Computational Grids

A

Thesis Report

Submitted in

Partial fulfillment of the requirement for the degree of

Bachelor of Technology in Computer Science

By

Nitin D. Sonkusale
and
Ajitabh Tiwari

Department of Computer Science Engineering
NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA
ROURKELA-769008 (ORISSA)
May 2009

National Institute of Technology Rourkela
Rourkela-769008 (Orissa)

Certificate

This is to certify that the work in this Thesis Report entitled "*Cost Based Optimization of job allocation in Computational Grids*" by ***Nitin D. Sonkusale*** and ***Ajitabh Tiwari*** has been carried out under my supervision in partial fulfillment of the requirements for the degree of *Bachelor of Technology* in Computer Science during session 2005-2009 in the Department of Computer Science and Engineering, National Institute of Technology Rourkela, and this work has not been submitted elsewhere for a degree.

Place: Rourkela
Date: May 11, 2009

(Bibhu. D. Sahoo)
Sr. Lecturer
Department of Computer Science
and Engineering

Acknowledgements

Many people have helped to create this thesis and each of their contribution has been valuable. We would like to express our gratitude to all those who gave us the possibility to complete this thesis. We want to thank the Department of Computer Science and Engineering for giving us an opportunity to commence this thesis and to do the necessary research work.

We are deeply indebted to our supervisor *Prof. B. D. Sahoo* whose help, stimulating suggestions and encouragement helped us in all the time of research for and writing of this thesis.

We are grateful to *Prof B. Majhi*, *Professor and Head, CSE* for his excellent support during our work. Thanks are due to all our classmates for their love and support.

Last, but not least we would like to thank all professors and lecturers for their generous help in various ways for the completion of this thesis.

(Nitin D. Sonkusale)
Roll number-10506036,
(Ajitabh Tiwari)
Roll number-10506002,
B.Tech.(Computer Science), 2005-2009

Abstract

Computational grids are distributed systems composed of heterogeneous computing resources which are distributed geographically and administratively. These highly scalable systems are designed to meet the large computational demands of many users from scientific and business orientations.

Grid computing is a powerful concept, its chief appeal being the ability to make sure all of a resource's computing power is used. In a grid world, the idle time of hundreds or thousands of resources could be harnessed and rented out to anyone who needed a massive infusion of processing power.

First, the architecture of a grid system is presented. The design gives a mathematical model of the grid system for efficiently allocating the grids resources. The challenges faced for optimal job allocation motivate the exploration in optimizing grid resource allocations. We have extensively surveyed the current state of art in this area. A grid server coordinates the job allocation for the grid users and helps to select the best resources for a job among different possible resource offers with the best prices offered.

Interaction between grid users and the resources require a mediator that uses different paradigm to communicate the needs of the two parties in terms of performance requirements, timing constraints, price charged etc. A game theoretic bargaining approach is studied to agree upon standard prices.

We have implemented various job allocation schemes in computational grids based on the mathematical modeling of the grid system and bargaining protocol with the objective function of optimizing the cost. The performance of the schemes have been analyzed and compared.

A new model for job allocation in computational grids has been proposed, for job allocation based on the clustering of resources.

Contents

List of Figures

Chapter 1 Introduction

In the past several years, grid computing has emerged as a way to harness and take advantage of computing resources across geographies and organizations. The recent explosion of commercial and scientific interest in the Grid makes it timely to revisit the question: What is the Grid, anyway?

This chapter is a brief introduction to grid computing.

1.1 The Reality

There has been a long running battle between centralized and distributed systems. Centralized systems have always been an area of research where we can concentrate more computational power into more confined and integrated space. On the other hand centralized systems provided with mostly compute intensive and challenging applications, constantly being one step ahead of the resources any centralized installation could provide. Thus each area is unique in its own way finding an application space that each can serve best.

However recent developments in the social, political and scientific areas have favored the emerging trends in the distributed paradigm. Following, or perhaps leading, international integration efforts, scientific research has moved from closed university campuses and governmental departments into a more cross-border collaboration effort, spanning many

countries, organizations and funding bodies. Money is being invested in the scientific research and infrastructure due to the stable economic conditions.

For a number of years, the economics of high performance distributed computing have been changing dramatically. Servers and storage have continued to rapidly improve their "price for performance" by leveraging new innovations and manufacturing efficiencies, and the same trend has finally taken hold for bandwidth. The effect is to transform distributed computing into a competitively priced commodity. At the same time, TCP/IP has become the only networking protocol suite considered, and UNIX (TM) or Linux has become the operating system of choice for scientific computing. In contrast to the commoditization of technology, skilled people have remained scarce, and concerns about security and quality of service have increased. The logical response to these changes is to move from a model based on discrete infrastructure components to a distributed computing model which fully leverages the computing capabilities of the infrastructure. The century saw a right set of enabling technologies and target markets inspiring new enthusiasm for distributed computing approaches.

1.2 Introduction to Grid Computing

Grid computing, adapted from Ian Foster is concerned with “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.” A grid is a decentralized heterogeneous system in which resources belong to multiple organizations.

Grid computing can mean different things to different individuals. From user’s perspective, a Grid is a collaborative problem-solving environment in which one or more user

jobs can be submitted without knowing where the resources are or even who owns the resources. Grid computing can be seen as a journey along a path of integrating various technologies and solutions that move us closer to the final goal. Its key values are in the underlying distributed computing infrastructure technologies that are evolving in support of cross-organizational application and resource sharing—in a word, virtualization—virtualization across technologies, platforms, and organizations. This kind of virtualization is only achievable through the use of open standards. Open standards help ensure that applications can transparently take advantage of whatever appropriate resources can be made available to them. An environment that provides the ability to share and transparently access resources across a distributed and heterogeneous environment not only requires the technology to virtualize certain resources, but also technologies and standards in the areas of scheduling, security, accounting, systems management, and so on.

There are three types of grids: computational grids, data grids and access grids. Computational grids have been the center of the researches on distributed computing. Allocating jobs to the various available resources in a grid has been a major area of interest. In this thesis, we compare various job allocation schemes and propose a new one, comparing them with the earlier ones.

1.3 Quality Of Service (QoS) in grid computing

It is openness and dynamics of the grid that bring about **QoS** problems peculiar to grid environment [3]. Openness gives rise to the heterogeneity of the grid and the multiplicity of services, users, and their demands. Dynamics, on the other hand, permits the grid resource, users,

and policies to change autonomously. All of this confronts us with great challenges and provides us with interesting issues in grid research.

When you deploy a grid, it will be to meet a set of business requirements. To better match grid computing capabilities to those requirements, it is useful to keep in mind some common motivations for using grid computing.

1.3.1 Exploiting underutilized resources

One of the basic uses of grid computing is to run an existing application on a different machine. The machine on which the application is normally run might be unusually busy due to a peak in activity. The job in question could be run on an idle machine elsewhere on the grid.

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines are busy less than 5 percent of the time over a business day. In some organizations, even the server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage.

1.3.2 Parallel CPU capacity

The potential for massive parallel CPU capacity is one of the most common visions and attractive features of a grid. A CPU-intensive grid application can be thought of as many smaller *subjobs*, each executing on a different machine in the grid. To the extent that these subjobs do not need to communicate with each other, the more *scalable* the application becomes.

A perfectly scalable application will, for example, finish in one tenth of the time if it uses ten times the number of processors.

1.3.3 Virtual resources and virtual organizations for collaboration

The users of the grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger grid. Files or databases can span many systems and thus have larger capacities than on any single system. Sharing is not limited to files, but also includes other resources, such as specialized devices, software, services, licenses, and so on. These resources are *virtualized* to give them a more uniform interoperability among heterogeneous grid participants.

1.3.4 Resource balancing

Occasionally, a project may suddenly rise in importance with a specific deadline. A grid cannot perform a miracle and meet a deadline when it is already too close to the deadline. However, if the size of the job is known, if it is a kind of job that can be sufficiently split into subjobs, and if enough resources are available after preempting lower priority work, a grid can bring a very large amount of processing power to solve the problem.

1.3.5 Reliability

High-end conventional computing systems use expensive hardware to increase reliability. They are built using chips with redundant circuits that vote on results, and contain logic to achieve graceful recovery from an assortment of hardware failures. The machines also use duplicate processors with hot pluggability so that when they fail, one can be replaced without

turning the other off. Power supplies and cooling systems are duplicated. The systems are operated on special power sources that can start generators if utility power is interrupted. All of this builds a reliable system, but at a great cost, due to the duplication of expensive components.

1.4 Literature Review

Aram Galstyan proposed Resource Allocation in the Grid Using Reinforcement Learning i.e. the only information that users receive is the expected completion time of a job it submits to a particular resource[10].

The use of many of the job allocation algorithms has been limited due to restriction in application designs, runtime system, or the job management system itself. Therefore simple allocation scheme such as First Come First Serve (FCFS) or is used in practice [4].

Current job scheduling practices typically support variable resource allocation to a job, and run to completion scheduling. Scheduling policies are also heavily based on First-come-First-serve (FCFS) methods [5]. A FCFS scheduling algorithm allocates resources to jobs in the order that they arrive. The FCFS algorithm schedules the next job in ready queue as soon as sufficient system resources become available to meet all of the job requirements. The advantage is that this provides level of determinism on the waiting time of each job [6]. The disadvantage of FCFS shows up when the jobs at the head of the ready queue cannot be scheduled immediately due to insufficient system resources, but jobs further down the queue would be able to execute given the currently available system resources. These latter jobs are essentially blocked from executing while the system resource remains idle.

Satish Penmatsa and Anthony T. Chronopoulos have given the mathematical modeling of the grid system and allocating jobs to the resources [1]. The system is based on a fair pricing strategy which is related to game theory by drawing upon the Nash Bargaining Solution [2].

1.5 Resource Allocation Problem in Computational Grid

In high throughput computing, the grid is used to schedule the independent job vs. dynamically distributed resources. The grid scheduling is used to utilize unused processors. The grid scheduling is to share resources in a large scale and to solve complex problem. In the grid environment the scheduler consider the set of jobs from the different users and set of available resources. The main objective of scheduler is to maximize the system utilization.

In general, the grid scheduling is divided into 3 phases. They are Resource Recovery, Scheduling, and Executing. In the second phase the grid servers find the best match between the set of jobs and available resources. The phase is proven to be an **NP-hard Problem**[8]. The computational grid is dynamic and has unpredictable behavior because:

1. Computational performance of each resource varies from time to time.
2. The Network connection may be unreliable.
3. The resources may join or leave the grid at any time.
4. The resource may be unavailable without a notification.

The two different goals for resource allocation are (1) high throughput, to minimize the execution time of each application, and (2) allocating a set of independent tasks to the resources to optimize the cost.

Thus, in this thesis we discuss job allocation schemes in computational grids based on cost optimization.

1.6 Performance to meet as per QoS

When a grid task is being submitted to a job management service, it is accompanied by a description of its required resources, such as the number of processors, the amount of memory and storage, and its estimated computation length. In addition, each task lists a number of allocation options, each of which reflects varied resources requirements resulting from task's malleability and the different resources on the grid. Each of the option's associated measures, known as the QoS metrics, is used to compute a utility value.

The utility value is a measure of the usefulness of an allocation option to the owner of both the tasks and the grid resources. As such, it depends on metrics of the computation task itself (including the length of the computation, the accumulated computation time, or the existence of a completion deadline) but also on metrics that directly reflect the perceived value that the owner of the task attaches to the task itself.

Thus the performances of the job allocation schemes depend on the utility value which is compared between the resource owners and the job allocating servers through bargaining to test the feasibility of allocating the jobs.

1.7 Approaches Used

The various job allocation schemes have their base in optimizing the cost of both the parties involved i.e. the grid users and the resource owners. These schemes can have a time-based approach or a price-based approach.

1.7.1 Time-based approach

Getting the jobs done in the minimum possible responsible time is the goal here. The job may be delayed due to delay in queuing, geographic locations of the resources, and network capabilities. An increase in allocation time causes loss to the users and to get the tasks done in minimum possible time has considerable overheads for the grid. These schemes are mainly applicable to cases where the time in which the jobs should be completed is critical.

1.7.2 Price-based approach

The jobs that are to be submitted by the users may have different execution costs based on their nature. Some jobs may require very high computing while others may be performed with moderate capabilities. On the other side, each resource owner has a fixed price for the resources to allow the execution of jobs. So this price between the two parties has to be balanced so that both sides agree on a price. These schemes apply to situations where the jobs require computing on large scales despite the time in which they are completed.

In this thesis we study and propose new job allocation strategies based on cost optimization in which the resource owners have standard prices and grid users have certain fixed prices and both have to agree on a mediated value.

1.8 Layout of the thesis:

In the first chapter we give a brief introduction to grid computing. The problem of job allocation in computational grids is discussed and a brief introduction is given about the approaches to handle job allocation. In the second chapter the grid computing structure is illustrated with its mathematical modeling. The job allocation schemes are discussed and we study two job allocation schemes based on the grid system model. Chapter three gives the pricing model and bargaining game theory model is discussed. In chapter the various results of our implementations are analyzed. In chapter five we propose a new approach based on clustering of the resources for job allocation.

1.9 Conclusion:

Computational grids are emerging as the preeminent model for large scale scientific computing. By providing access to heterogeneous resources distributed across administrative domains, grids provide simplified access to vast computing resources. The allocation of jobs to these resources has been a area of interest in the recent past. There two different goals for resource allocation is high throughput and minimizing the cost. In this thesis we discuss job allocation schemes in computational grids based on cost optimization.

Chapter 2 Grid Computing Structure

Here the basic grid system model is presented. A grid system tries to solve problems submitted by various grid users by allocating the jobs to the computing resources governed by different resource owners.

2.1 System Components

The various components of a grid computing system are described below.

2.1.1 Jobs and applications

Although various kinds of resources on the grid may be shared and used, they are usually accessed via an executing *application* or *job*. Usually we use the term *application* as the highest level of a piece of work on the grid. However, sometimes the term *job* is used equivalently.

2.1.2 Grid community

The users who want their jobs to be executed which require high compute capacities are a part of this community. These users never actually get to see the * where actually their jobs are being submitted. The system is virtual.

2.1.3 Grid controller

It acts on behalf of the grid community. It is responsible for assigning the jobs to the grid servers.

2.1.4 Grid servers

These are specialized systems which collect the jobs coming from the grid controller. They are responsible for allocating the various jobs to the resources and bargaining the costs in exchange of the computations performed on the resources. Thus these servers are also responsible for job allocation and scheduling the jobs to the resources available on the network based on cost optimization.

2.2 Queuing system

The system has **m** grid servers and **n** computers.

Jobs arrive on the grid server as submitted by the grid controller on behalf of the grid community. Each computer is modeled as an M/M/1 queuing system. Thus we have the following configuration:

- Poisson distribution of the arriving jobs
- exponentially distributed processing times
- number of servers is 1

Let the total job arrival rate be φ.

Let the job arrival rate at each server be φ_j.

Hence, we have:

$$\varphi = \sum_{j=1}^{m} \varphi_j$$

The average processing rate of each computer be $\mu_i,\ i = 1, 2, 3, \ldots, n.$

The total job arrival rate φ must be less than the aggregate processing rate of the system,

$$\text{i.e. } \varphi < \sum_{i=1}^{m} \mu_i$$

Each server keeps track of the price per unit resource vector p_{ji} (the bargaining game is played prior to the job allocation) and the processing rate μ_i of the i^{th} computer. Since each grid user will have a different reserved valuation, the p_{ji} depends on the user on whose behalf the server j plays the game with the i^{th} computer and so the server has to maintain separate price vectors for each grid user.

2.3 Modeling the system

The overall model of the grid can be presented as given in the figure 3.1. The jobs are submitted to the various available resources in the grid.

Let s_{ji} denotes the load fractions of each server j (j= 1, 2, …, m) that are assigned to computer i ($\sum_{i=1}^{n} s_{ji} = 1$ and $0 < s_{ji} < 1, i = 1, 2, \ldots, n$) such that that the expected price of all the jobs in the system is minimized.

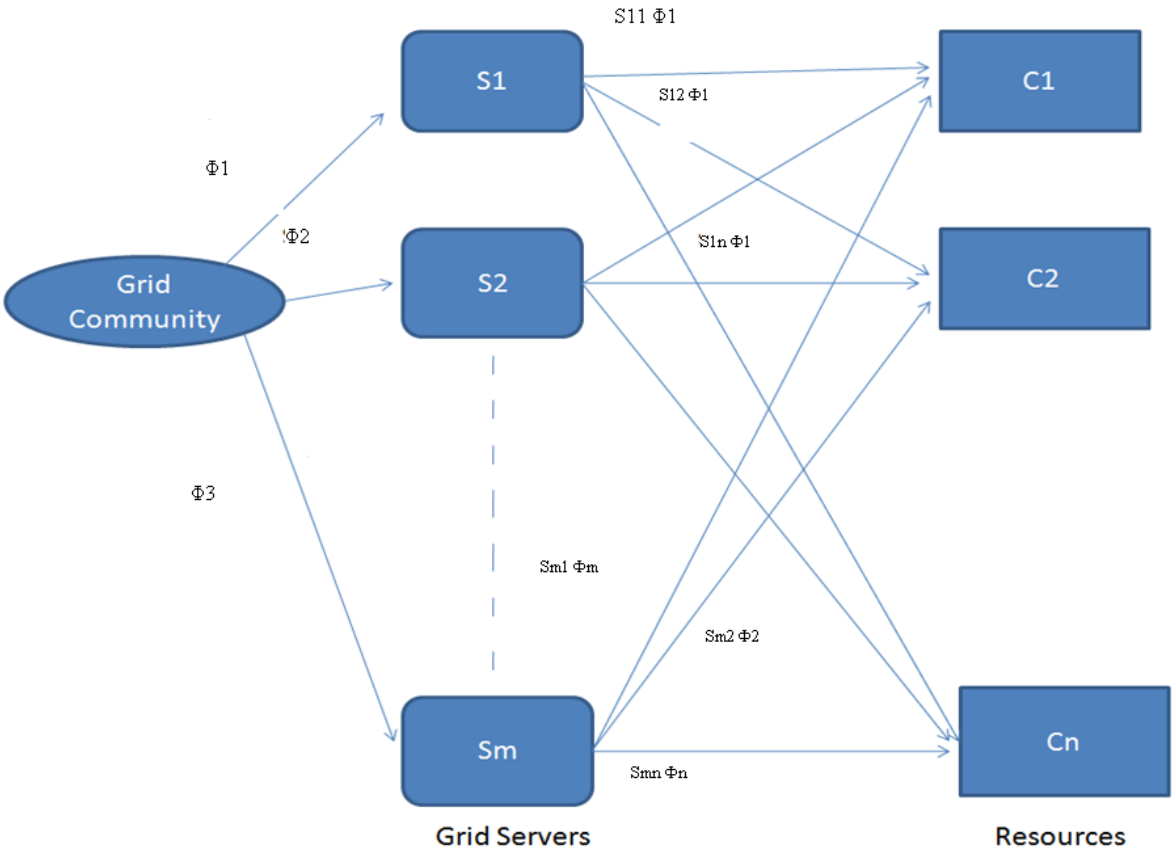

Figure 2.1 Grid system model.

Let s_{ji} be the fraction of workload (jobs) that server j sends to computer i. Thus, $s_j = (s_{j1}, s_{j2}, s_{j3}, \ldots, s_{jn})$ denotes the workload fractions of server j and the vector $s = (s_1, s_2, \ldots, s_m)$ denotes the load fractions of all the servers.

Since each computer is modeled as an M/M/1 queuing system, the expected response time at computer i is given by:

$$F_i(\mathbf{s}) = \frac{1}{\mu_i - \sum_{j=1}^{m} s_{ji}\phi_j} \qquad (2.1,[1])$$

Thus the overall expected cost of server *j* is given by:

$$D_j(\mathbf{s}) = \sum_{i=1}^{n} k_i p_{ji} s_{ji} F_i(\mathbf{s}) = \sum_{i=1}^{n} \frac{k_i p_{ji} s_{ji}}{\mu_i - \sum_{k=1}^{m} s_{ki}\phi_k} \qquad (2.2,[1])$$

k_i is assumed to be a constant which maps the execution time to the amount of resources consumed at node.

p_{ji} is the agreed price as a result of the bargaining game between server j and computer i.

The overall expected cost of the system (i.e. of all the servers) is given by:

$$D(\mathbf{s}) = \frac{1}{\Phi} \sum_{j=1}^{m} \phi_j D_j(\mathbf{s}) \qquad (2.3,[1])$$

which is equivalent to,

$$D(\mathbf{s}) = \frac{1}{\Phi} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{k_i p_{ji} \phi_j s_{ji}}{\mu_i - \sum_{k=1}^{m} s_{ki}\phi_k} \qquad (2.4,[1])$$

subject to the constraints:

$$s_{ji} \geq 0, \qquad i = 1, \ldots, n, \qquad j = 1, \ldots, m \qquad (2.5[1])$$

$$\sum_{i=1}^{n} s_{ji} = 1, \qquad j = 1, \ldots, m \tag{2.6,[1]}$$

$$\sum_{j=1}^{m} s_{ji}\phi_j < \mu_i, \qquad i = 1, \ldots, n \tag{2.7,[1]}$$

Thus the system is formulated as a mathematical problem of allocating the jobs by the servers to the various resources based on the above objective function i.e. equation 4. The constraints are given by equations 2.5, 2.6 and 2.7.

Based on the above formulation of the problem there are different job allocation schemes as discussed in the next section.

2.4 Job allocation schemes

In a Grid Scheduler, the mapping of Grid resources and an independent job in optimized manner is so hard where we couldn't predict optimized mapping. The aim is to investigate effective resource allocation techniques based on computational economy.

2.4.1 Intra-Site Job Execution Strategies

This problem concerns about the strategies of the participating computers inside a Grid site. Specifically, although the various computers participate in the making up of the Grid site, each individual computer is selfish in that it only wants to execute jobs from local users but does not want to contribute to the execution of remote jobs.

2.4.2 Intra-Site Bidding

This problem concerns about the determination of the advertised "execution capabilities" for jobs submitted to the global scheduler. For the scheduler to allocate jobs using a certain scheduling algorithm, it needs to know all the sites' execution capabilities i.e. these are modeled as the execution times needed for the pending jobs. To determine the execution time needed for a certain job, within a Grid site each participating computer can make a "declaration"—a notification to the local job dispatcher specifying the time needed to execution the job.

2.4.3 Inter-Site Bidding:

Similar to the intra-Site situation, at the inter-site level, the various local job dispatchers also need to formulate game theoretic strategies for computing the single representative value of the job execution time to be sent to the global scheduler.

2.5 Price-based job allocation schemes

There are two job allocation schemes for a particular grid system model based on an existing pricing model. The two schemes differ in their objective. One tries to minimize the cost of the grid community (i.e. grid users) by taking the jobs at all the grid servers into account whereas the other tries to minimize the cost of the grid users by taking the jobs at each grid server independently of the others.

2.5.1 Global Optimal Scheme with Pricing

This scheme tries to minimize the cost of all the jobs in the system (i.e. jobs at all the servers). The jobs assigned to a computer by a grid server are processed completely by it and are

not transferred any further. The problem is formulated as a cost minimization problem and provides a solution.

The load fractions (*s*) are obtained by solving the nonlinear optimization problem D(s) (eq. 2.4) which gives the optimum expected cost of the system. To find the solution, the scheme finds the load fractions of each server by taking into account the load on each computer due to the other server allocations.

2.5.2 Nash Scheme with Pricing

NASH distributed load balancing scheme [7] includes pricing where each grid server tries to optimize its objective function (minimizing the cost) independently of the others and they all eventually reach equilibrium. In general, the jobs from a grid user will be dealt by a local server and so this scheme is favorable to the individual users but not to the entire system. This situation can be viewed as a non-cooperative game among the servers. The equilibrium is called *NASH Equilibrium.*

The NASH algorithm for load balancing in distributed systems [7] is modified to include pricing. In this scheme each server tries to minimize the total cost of its jobs independently of the others. The load fractions are obtained by formulating the problem as a non-cooperative game among the servers. The goal of server j is to find a feasible job allocation strategy S_j such that D_j (s) (eq 2.2) is minimized.

2.6 Conclusion:

The grid users who wish to make use of the vast computing pool available in the grids have to can do so by submitting their jobs to the grids servers. These servers act as mediators they allocate the jobs to the resources. Both the parties communicate their prices to these servers and a price negotiable among the entities is reached. The grid system can be modeled mathematically and based on this model job allocation schemes are proposed.

Chapter 3 Resource Allocation Using Bargaining Game

A grid system tries to solve problems submitted by various grid users by allocating the jobs to the computing resources governed by different resource owners. The prices charged by these owners are obtained based on a pricing model using a bargaining game theory framework. These prices are then used for job allocation.

3.1 The Idea

Game Theory approaches have been proposed earlier to develop economic models for resource management and scheduling in Grid computing. However, this work suffers from lack of formulation in the sense that the actual mappings of the problem at hand into a game between two players has not been shown, nor are stated analytical modeling and results, which make the problem more difficult to solve in real life.

Players:

1] the grid server acting on behalf of the grid community

2] the computing resources

These two play an incomplete information alternating-offer non-cooperative bargaining game to decide upon the price per unit resource charged by that mobile device.

3.2 Nash Equilibrium

In game theory, the Nash equilibrium is a solution concept of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only his or her own strategy unilaterally. This concept is used to analyze the outcome of the strategic interaction of several decision makers. In other words, it is a way of predicting what will happen if several people or several institutions are making decisions at the same time, and if the decision of each one depends on the decisions of the others.

So this equilibrium strategy is applied to the bargaining game between the grid servers and the resources.

The concept of incomplete information ensures that the two players have no idea of each other's reserved valuations, i.e., the maximum offered price for servers (acting as the buyer of resource) and minimum expected price for computers (acting as the seller of resources).

Assuming there are n computers under a single grid server, the server has to play n such games with the corresponding devices to form the price per unit resource vector, p_i. In particular, by drawing upon the Nash bargaining framework from non-cooperative game theory, the pricing strategy is guaranteed to be fair.

3.3 Game Notation for Pricing Model

Assume there are **m** grid servers and **n** computers. At time **t**, let the number of devices under Server **S*i*** be denoted as $p(t)i$, where $1 \leq i \leq m$.

Clearly, $\sum_{i=1}^{P} p(t)i = \text{n}$.

The scenario can be modeled as a one to one bargaining relationship between the computer and its current grid server as a bargaining game, BGj for $1 \leq j \leq n$. So, in a system with **m** servers and **n** computers at time t, we have **m*n** bargaining games.

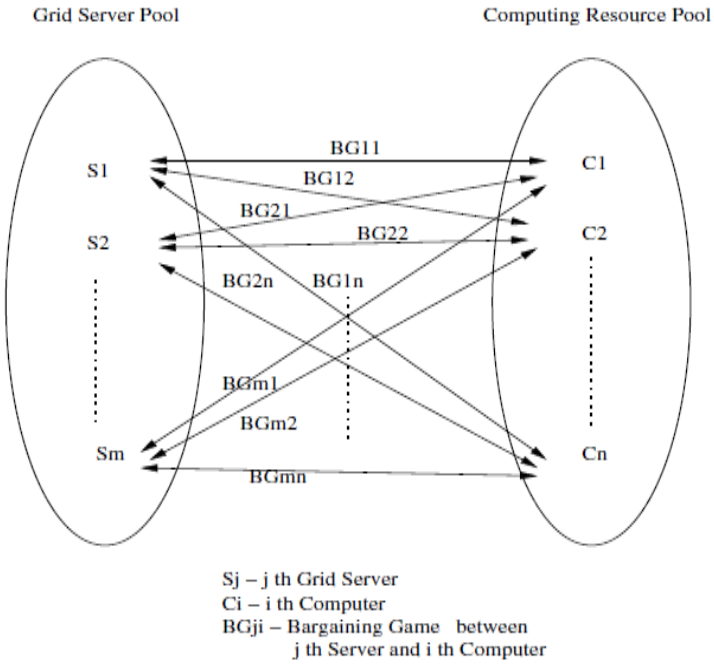

Figure 3.1 Bargaining game mapping between the grid servers and computers.

Both the players will try to maximize their utility functions which are defined later between the two players and thus the game reduces to the simple case of dividing the difference of maximum buying price offered by the grid community and minimum selling price expected by the mobile users (called the reservation values according to game theory conventions). A complete information cooperative bargaining theory solution requires both players to know these reservation values beforehand, which is not realistic, particularly for a grid environment. Therefore, the game is modeled using incomplete information alternating offers bargaining theory.

3.4 The Bargaining Protocol

The bargaining procedure is as follows. Bargainer A starts the negotiation by sending a proposal to bargainer B. Now B can either accept or reject it. If the offer is accepted, then the bargaining ends and the agreement is implemented. If the offer is rejected, then B must send back a counterproposal to specify his preferences to A. Now A will evaluate the proposal and choose either to accept or reject it. This process continues until an agreement is reached.

One of the players starts the game. Let the server starts the game:

Server - propose an offer which is much less than its own reserved valuation

if offered price >= the computer's standard price with highest expected surplus
then computer accepts the offer
else it makes a counter offer

if this offered price <= the server's standard price with the highest expected surplus
then accepts
otherwise it offers again

This procedure continues until an agreement is reached.

At each step, the expected surplus of each player is based on the probability of acceptance, breakdown or counter-offer of the other player. In general, they are given by:

Expected Utility = E[Surplus]
= (reserved valuation of x - standard price of x) - probability(standard price) [9]
where x stands for grid server or the computer

probability (standard price) is the probability that the standard price will be accepted by the other player as predicted by itself

standard price represents the different offered prices used by the players to compute their expected surplus

At each step, if an offer is rejected, then the players will update (i.e. reduce) the *probability (standard price)* which monotonically decreases as the alternatives come closer to their reserved valuations where it is more likely to be accepted by the opponent .

As the time increases, the expected surplus gradually decreases, which make both the players, offer prices which are much closer to their reserved valuation and which helps the game to converge.

3.5 Conclusion

The grid system is a collection of grid servers and computers (i.e. resources). These servers try to find the resources on which the jobs from various users can be executed. The negotiation between these two entities is formulated as an incomplete information alternating-offer non cooperative bargaining game in [4] with the grid servers playing on behalf of the grid users.

Chapter 4 Simulation and Results

4.1 Job allocation Scheme 1:

Attaining Equilibrium after Independent Cost Optimization

In this job allocation scheme each grid server tries to optimize its objective function (minimizing the cost) independently of the others and they all eventually reach equilibrium. The server tries to minimize the total cost of its jobs independently of the others. The load fractions are obtained by formulating the problem as a non-cooperative game among the servers. The goal of server *j* is to find a feasible job allocation strategy s_j such that $D_j(s)$ in equation 3.2 is minimized.

Equation 3.2

$$D_j(\mathbf{s}) = \sum_{i=1}^{n} k_i p_{ji} s_{ji} F_i(\mathbf{s}) = \sum_{i=1}^{n} \frac{k_i p_{ji} s_{ji}}{\mu_i - \sum_{k=1}^{m} s_{ki}\phi_k}$$

Algorithm:

Step 1: Sort the resources in decreasing order of their available processing rates.

Step 2: Calculate the processing rate per unit price for each resource and store it in S_{avg} =

$$\left(\frac{\mu_1^j}{\sqrt{\mu_1 k_1 p_{j1}}} \geq \frac{\mu_2^j}{\sqrt{\mu_2 k_2 p_{j2}}} \geq \ldots \geq \frac{\mu_n^j}{\sqrt{\mu_n k_n p_{jn}}}\right);$$

Step 3: Sort S_{avg} in decreasing order.

Step 4: Some of the resources in the grid can be discarded by checking their processing rate per unit price ($S_{avg\,i}$) against a threshold value, which is given by:

$$t = \frac{\sum_{i=1}^{n} \mu_i^j - \phi_j}{\sum_{i=1}^{n} \sqrt{\mu_i p_{ji} k_i}}$$

Step 5: For each resource who has its processing rate per unit price ($S_{avg\,i}$) less than or equal to the threshold value **t**, then the server allocates $S_{jn} = 0$ (load fraction) to the corresponding resource.

Step 6: Calculate a new threshold t by ignoring the resource which was not allocated any load fraction.

Step 7: Repeat steps 4, 5 and 6 till processing rate per unit price ($S_{avg\,i}$) is greater than the threshold value.

Step 8: The load fractions are allocated to each of the remaining resources given by

$$s_{ji} = \frac{1}{\phi_j}\left(\mu_i^j - t\sqrt{\mu_i p_{ji} k_i}\right)$$

where S_{ji} is the fraction of workload (jobs) that server *j* sends to computer *i*.

4.2 Job allocation Scheme 2:

Global Optimal Scheme with Pricing

In this scheme we make a slight modification to the scheme described above in order to obtain the load information from the other servers and compute the available processing rates at each resource.

We have implemented an FCFS job submission policy by the grid community to the servers.

The load fractions (s) are obtained by solving the nonlinear optimization problem D(s) (equation 3.4) which gives the optimum expected cost of the system. To find the solution, the scheme finds the load fractions of each server by taking into account the load on each computer due to the other server allocations.

Algorithm:

Step 1: For server j we find the load fraction by using the previous job allocation scheme.

Step 2: For each resource i we update the available processing rate as seen by server *j*+1 which is given by

$$\mu^{i}_{j} = \mu_i - \sum_{k=1,k\neq j}^{j} s_{ki}\,\Phi_k$$

Step 3: If the available processing rate of any of the resource $\mu_j^i = 0$ then the corresponding resource is discarded while calculating the load fractions in the above steps.

Step 4: Repeat the above two steps for each server

We have also implemented FCFS and Random job allocations schemes to compare the results of the two schemes given above

4.3 FCFS Job allocation

The server allocates jobs as they arrive to the resources by searching the resources which available at that time. If a complete job cannot be allocated then the fractions are allocated depending on the available processing rates.

Algorithm:

Step 1: For each server *j*, allocate the first available resource by checking its processing rate as against that required by the job.

Step 2: Update the processing rates of each resource by subtracting the allocated rate to a job.

Step 3: As the jobs arrive at the server the resources are checked sequentially for their available processing rates. The load fractions are allocated depending on the available rate.

Step 4: The above steps are repeated till the job is completely allocated.

4.4 Random Job allocation

The server allocates the jobs randomly to the resources available.

Algorithm:

Step 1: For server j, allocate the jobs randomly to any of the resources. If the available resource cannot fully execute the task, allocate fraction of the load.

Step 2: As the job fraction gets allocated to the resources their available processing rates are updated.

Step 3: In this manner the jobs allocated to the available resources. If at any particular resource selected by the server cannot fulfill the job requirement, the next resource is randomly selected.

4.5 Experimental results

In this section the results of our experiments and comparison among the various schemes are illustrated

4.5.1 Simulation environment

We simulated the performance of our job allocation schemes in MATLAB version 7.0. The main performance metrics used in our simulations is the price per system utilization for all the algorithms. As the no of resources is always greater than the servers in a grid environment, it is always the case that many of the resources available may not be utilized or may be ideal thus decreasing the overall system utilization. But as we try to increase the system utilization the total cost of the system increases. So, there is a tradeoff between the overall system utilization and the total cost of the system.

4.5.2 Performance Evaluation

We evaluated the schemes presented above under various system loads and configurations. In the following we present and discuss the simulation results.

4.5.2.1 Effect of system utilization

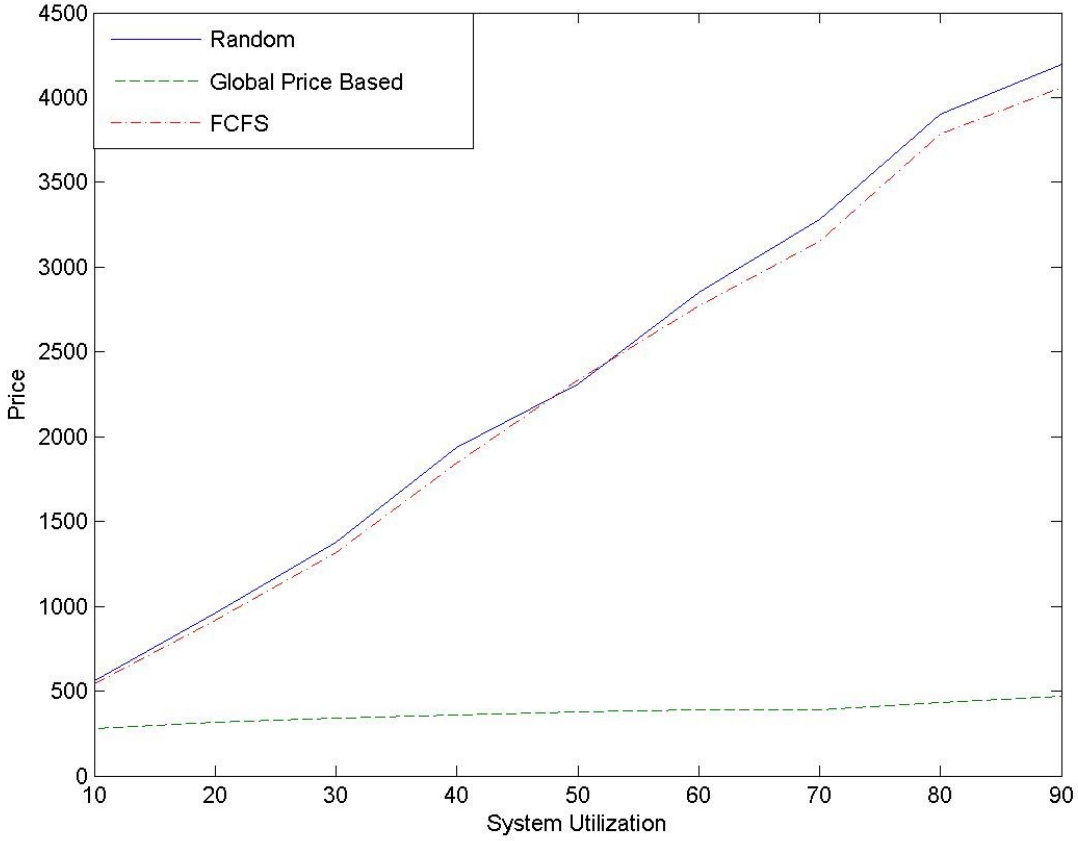

Figure 4.1 System Utilization v/s Price

To study the effect of system utilization we simulated a heterogeneous system consisting of 32 computers with eight different processing rates. This system is shared by 20 servers.

For each experiment the total job arrival rate in the system Φ is determined by the system utilization and the aggregate processing rate of the system. System utilization *is* defined as the ratio of the total arrival rate to the aggregate processing rate of the system.

We choose fixed values for the system utilization and determined the total job arrival rate Φ.

As seen from the above graph in fig 4.1, for all the three algorithms FCFS, Random and Global optimization scheme using Pricing, as the system utilization increases, the price increases. This is supported by the reason that as the system utilization increases, the number of resources utilized also increases thus increasing the cost of the system. Also, from the above graph we can easily conclude that the Global optimization scheme is better than the FCFS and Random algorithm when the price that user has to pay is taken into consideration. This can be supported by the plot of Agreed price between a server and resource and the Workload fraction using this scheme fig 4.2

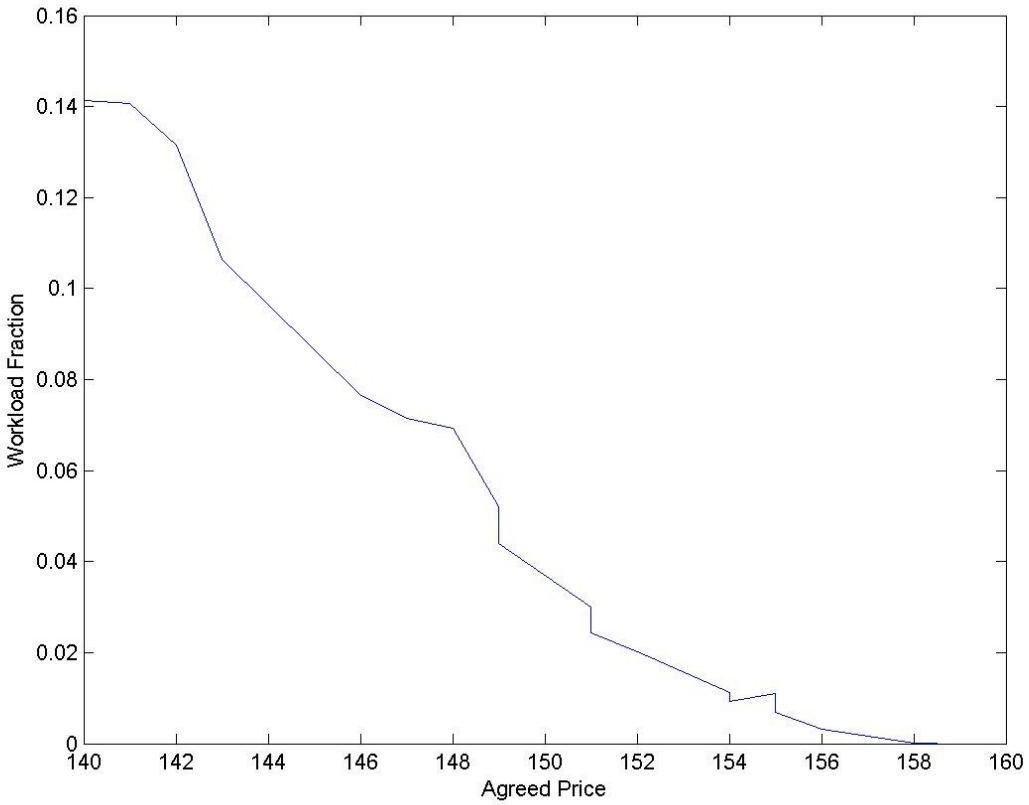

Figure 4.2 Agreed Price v/s Workload Fraction

4.5.2.1 Effect of heterogeneity

In a grid, heterogeneity usually consists of processor speed, memory and I/O. A simple way to include heterogeneity is to have resources with different processing speed. The effect of having heterogeneous resources is shown by the above graph.

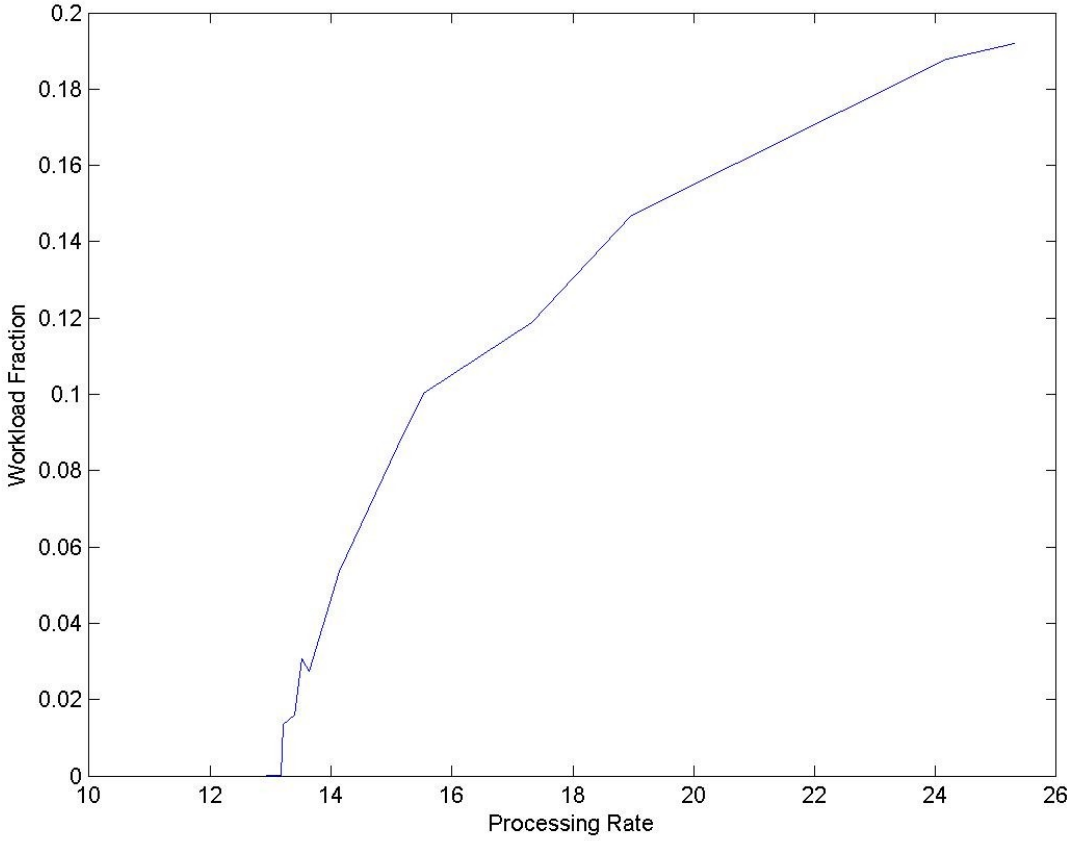

Figure 4.3 Processing Rate vs Workload fraction allocated.

As seen from the above graph as the processing rate of the resources is increased, the workload fraction allocated to them is increased.

Another example where heterogeneity comes to play is expected response time of the system.

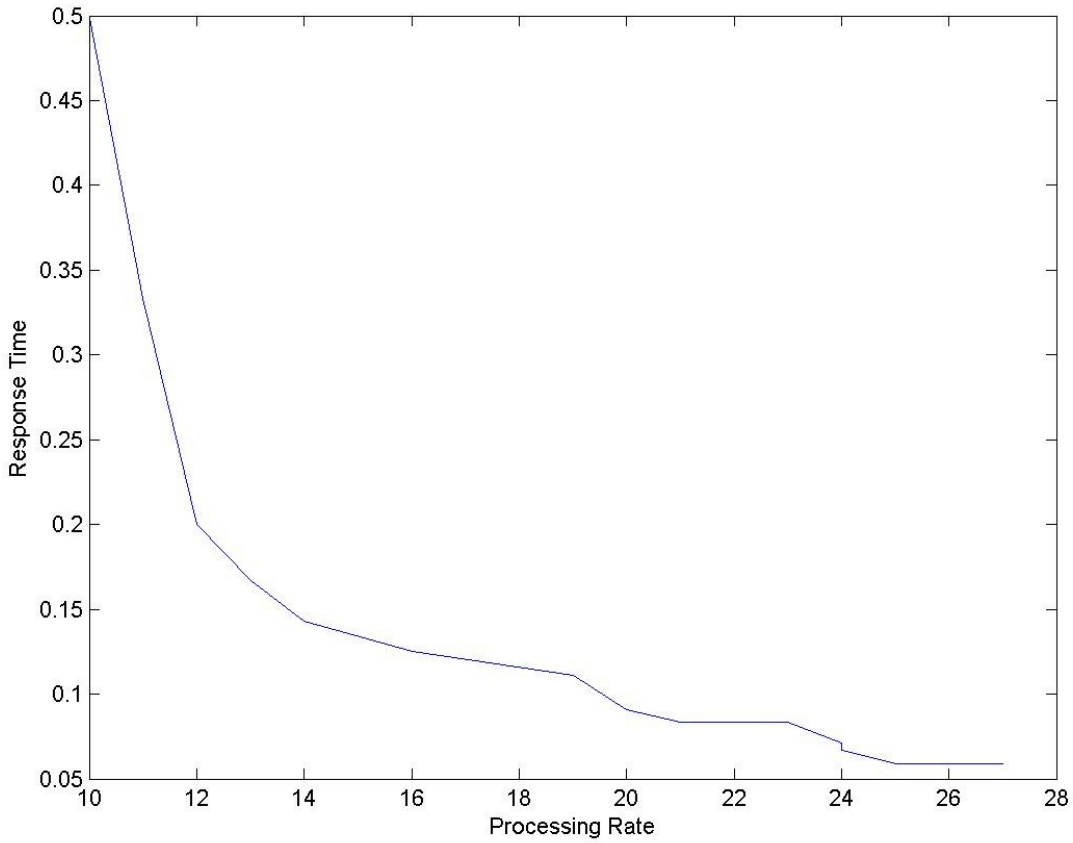

Figure 4.4 Processing Rate v/s Expected Response time

Figure 4.4 plots the expected response time with increasing processing rate. As expected the response time decreases as the Processing rate of resources increases.

4.6 Conclusion

Two job allocation schemes based on pricing for computational grids have been implemented. These schemes are formulated as constraint minimization problem. The algorithms to compute the optimal load (job) fractions for the grid servers are devised. As the system utilization increases the price charged by the resources increases leading to an overall increase in the execution of a job. The global optimal scheme with pricing based on the grid system model discussed in the chapter two provides the minimum cost among the FCFS and random schemes.

Chapter 5 Future Enhancements

With significant recent developments in creating the infrastructure for grid computing, the transparent sharing of resources at multiple geographically distributed sites is being facilitated. An important aspect of significance to multi-site job allocation is that of heterogeneity–different sites are generally unlikely to have identical configurations of their processors and can be expected to have different performance characteristics. Much of the research to date on job scheduling for heterogeneous systems has only addressed the scheduling of independent sequential jobs where each task is sequential.

We address the problem of heterogeneous multi-site job allocation, where each site hosts a homogeneous cluster of processors and propose clustered job allocation method.

5.1 Introduction

A computational grid consists of different types of resources with different owners. Usually those resources are not exclusively dedicated to grid usage. Sites can freely participate in grid computing by offering resources. The interaction between those grid resources during the execution of a job requires special servers that use a different job allocating paradigm than that of local or centralized allocators.

While local scheduling involves usually a single scheduling instance that has access to all system information, grid scheduling requires interaction to remote sites and their local

scheduling systems. This leads to the use of several scheduling layers for the grid and those layers need to exchange information.

The grid job scheduler needs:

- Information about available resources.
- Available features of the local level scheduler.

When the grid job scheduler will receive user request, it searches for the resources available for that task by maintaining a status table. The task may have a number of subtasks and some of those may be executed concurrently. If several resources are capable to fulfill a request, the grid job scheduler can first query those systems for the allocation and afterwards make its decisions to accept the allocation.

Resources may be present at a single site or multiple sites. The Grid job scheduler will first call the local scheduler at remote sites for getting the information about the status of resources available at that site. If the available resources can satisfy the job request, then the job will be executed otherwise, it will reject the request to that site. If any one of the available sites cannot satisfy the request, then either it will be executed locally or will be rejected.

The cluster computing system is referred as a "local scheduler", as compared to a “global” Grid scheduler. Grid scheduling is the process of scheduling applications over Grid resources. A Grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and usually owns the resource. A Grid scheduler is in charge of

resource discovery, Grid scheduling (resource allocation and task scheduling), and job execution management over multiple administrative domains. We focus our work on resource allocation.

The grid scheduler communicates with the cluster scheduler to obtain information about resource availability. In most cases, the grid scheduler maintains its own job queue and routes jobs to the various clusters participating in the grid using the services of a grid.

In a cluster, a job consists of a request for resources, a required execution environment, and one or more credentials indicating who is running the job. In a grid environment, there are also extensions to the standard job allowing a request or to specify a subset of clusters to select, co-allocation constraints, and other attributes.

All grid scheduling systems work on the basis that the "new task" which wants to be executed has to make itself known to a "resource selector." In current systems, the resource selector acts as a gateway to the grid. It will select resources from a global directory and then allocate the job to one of the available grid nodes. Typically, job allocation is done in two stages. Firstly a job is allocated to a particular node on the grid and then within that node, the job will be scheduled onto the processor.

5.2 Framework for Grid System

It is assumed that there exists a collection of heterogeneous systems formed as mini-grids, which are under the control of local scheduler. These heterogeneous systems own

resources, which are transparent to the local scheduler. All these local schedulers are under the control of a Global (Grid) Scheduler.

Suppose a task is to be processed at a node, it checks for its own resources for that particular task. If they are found, it executes the task. Otherwise it submits the task to the Grid Scheduler. Grid Scheduler has the current status of all the systems under its control. This status information is stored in a table. Depending on the availability of the system resources, it groups (clusters) the systems in the form of mini-grids. Then the global scheduler assigns the task to an appropriate local scheduler [Figure 5.1].

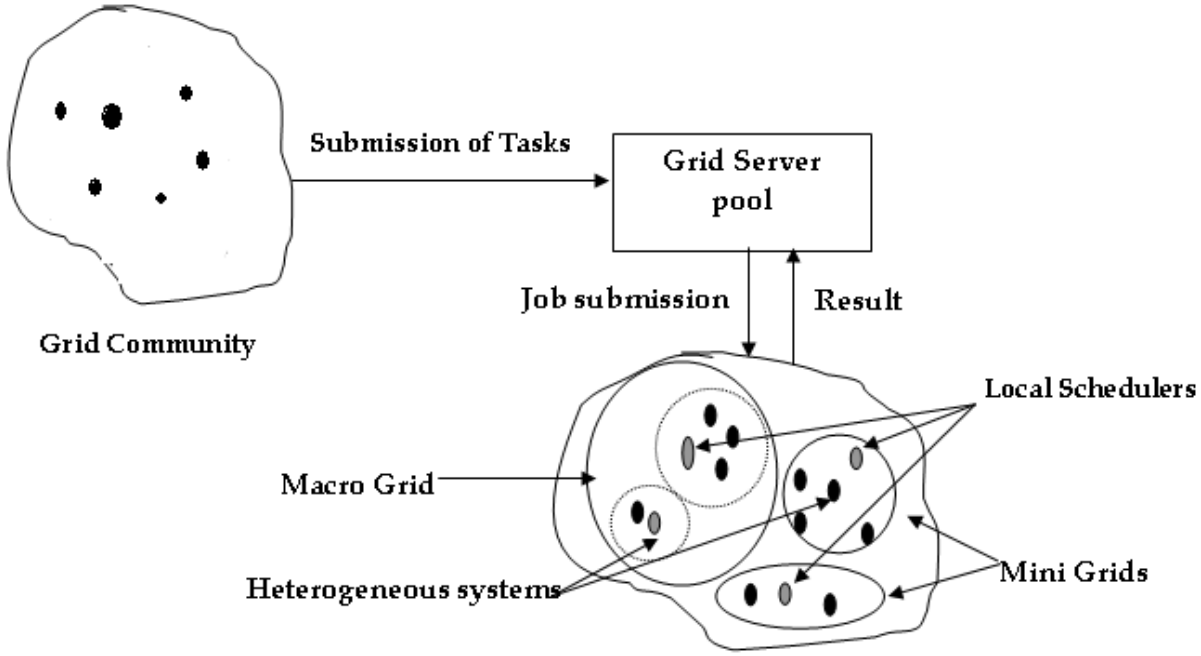

Figure 5.1.Model for Grid Scheduler

Resources are grouped so as to achieve homogeneity in a particular group. This grouping can be based on many parameters like the computing capabilities of the resources, their location,

the level of communication amongst the resources etc. When the jobs are submitted to the grid servers by the grid community these servers allocate these tasks to the appropriate resource group. This is done by checking the table that each server has which stores the information of the various sub-grids about the kind of resources each group has. After this decision is made of allocating the job to a local scheduler the task is allocated to the respective group.

At each local server the job allocation schemes discussed to allocate the various the job to the resources in a group can be efficiently employed.

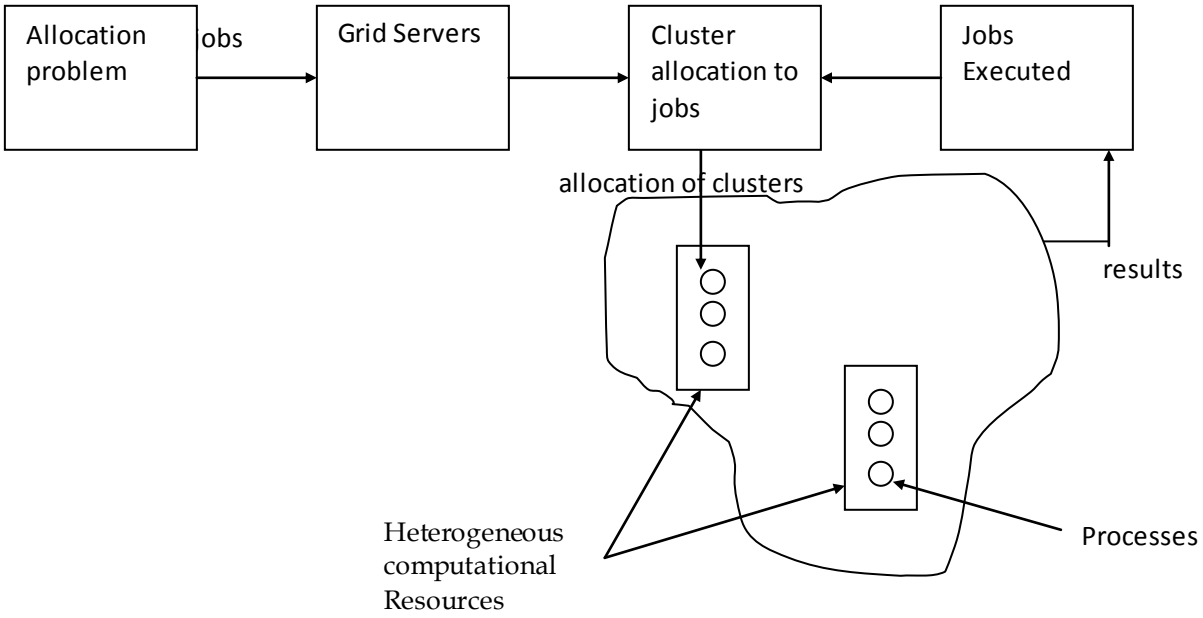

Figure 5.2 Representation for job allocation using resource clustering technique

5.3 The approach

This is a two phase process. In the first phase the resources are grouped together based on some criteria (like processing power) and appropriate group is selected by the grid server for a

job depending on its nature when a job arrives. In the second phase the job allocation schemes that we discussed in the previous chapter is applied.

5.3.1 First Phase: Resource Clustering

The first phase of the proposed approach consists of grouping the resources in terms of their computing capacities. We assume that the resource managers have knowledge of the communication cost with the other members of the grid. The resources are grouped according to the computing capabilities of the resources.

The computational grid can be modeled as a tree of resource clusters, which are geographically distributed. Each cluster is under single administrative control, called a local sub-grid server, who provides the current view of the cluster to the scheduler.

The grid resources are represented by a weighted undirected graph $G_p = (V_p, E_p)$, which we refer to as the system graph. It consists of a set of vertices $V_p = \{p_1, p_2, \ldots, p_n\}$, and a set of edges $E_p = \{(p_i, p_j) \mid p_i, p_j \in V_p\}$. Each processor (vertex) p_i has a processing weight w_i, modeling its processing cost per unit of computation. Each link (edge) has a link weight $c_{i,j}$, that denotes the communication cost per unit of communication between processors p_i and p_j

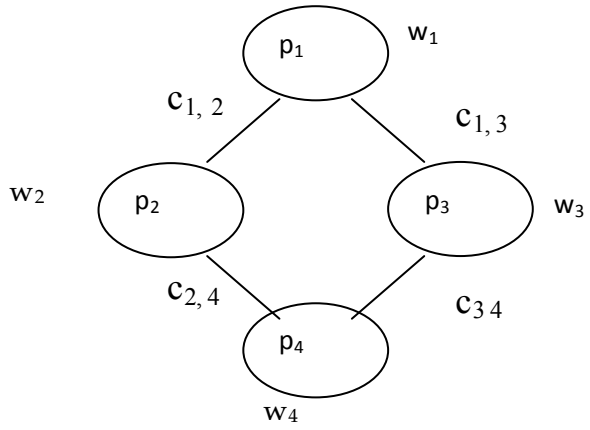

Figure 5.3 Graph representation for resources in a grid

Resources can be clustered by constructing sub-trees out of the computational grid tree modeled as a tree. The resources in a group can be a part of a sub-tree.

5.3.2 Second Phase: Job Allocation

Once the resources are clustered in homogeneous groups then we can apply the job allocation schemes that we have implemented. When the jobs arrive, the grid servers search for the resource pool depending on the nature of the job. When the resource group gets selected then the jobs are allocated using the job allocation schemes on the particular group of resources.

5.4 Conclusion

A new model for job allocation in computational has been proposed. Incorporating clustering of the resources into the job allocation problem can be a significant improvement in the grid computing area of research. In a Grid environment which consists of a large number of resources, different administrative domains may set different resource usage policies. The grid

servers can locate appropriate resource domains which are grouped together. After this the job allocation schemes implemented in the previous chapters can be effectively applied.

Chapter 6 Conclusion

This thesis has presented a novel approach to the allocation of grid resources. The approach presented applies techniques to improve upon existing grid resource allocation strategies. To efficiently utilize the computing capabilities of the grid there is a need for proper communication among the grid users and the resource owners regarding the prices that both parties can agree upon. In this thesis we have studied the mathematical modeling of the grid system. We have focused on the minimization of cost for both the users and the resources through a bargaining game in which each of the grid server and the resource try to agree upon a price against its standard evaluations. Job allocation schemes are implemented by formulating them as a cost minimization problem. The schemes are compared against the FSFS and Random job allocation schemes under simulations with various system loads and configurations. These schemes compute the optimal load fractions for the grid servers. The Global optimal scheme with pricing is better than the FCFS and Random algorithm when the price that user has to pay is taken into consideration. A new approach based on resource grouping is proposed for grids encompassing huge number of resources and so that the job allocation schemes implemented can be applied in such environments.

Chapter 7 References

[1] S. Penmatsa and A. T. Chronopoulos, Job Allocation Schemes in Computational Grids based on Cost Optimization, Proc. of the 19th IEEE Intl. Parallel and Distributed Processing Symposium, Joint Workshop on HPGC and HIPS, Denver, Colorado, 2005.

[2] P. Ghosh, N. Roy, K. Basu, and S. Das. A game theory based pricing strategy for job allocation in mobile grids. In *Proc. of the 18th IEEE International Parallel and Distributed Processing Symposium*, pages 26–30, Santa Fe, New Mexico, USA, 2004.

[3] Pu Juhua, Xiong Zhang and Wu Zhenxing. The research on QOS for grid computing in *Proceedings of ICCT2003*

[4] Vijay Subramanian, Rajkumar Kettimuthu, et al, "Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests", *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, 2002. HPDC-11 2002. Pages: 359- 366

[5] C.Bitten, J. Gehring, et. Al, "The NRW - Meta Computer: building block for a worldwide computational Grid", *proceeding of the 9th Heterogeneous Computing workshop*, pp.31-40, 2000

[6] C.Ememann, V. Hamscher, et. al, "On Advantageous of Grid Computing for parallel job scheduling", *proceeding 2nd IEEE/ACM Int'l Symp. On cluster- computing and the Grid (CCGRID 2002)*, Berlin, 2002, IEEE press.

[7] D. Grosu and A. T. Chronopoulos. A game-theoretic model and algorithm for load balancing in distributed systems. In *Proc. of the 16th IEEE International Parallel and Distributed Processing Symposium*, pages 146–153, Ft Lauderdale, Florida, USA, 2002.

[8] Fernandez-Bacad " Allocating modules to processors in a distributed System", IEEE transaction on Software Engineering

[9] P. Winoto, G. McCalla, and J. Vassileva. An extended alternating-offers bargaining protocol for automated negotiation in multi-agent systems. In *Proc. of the 10th International Conference on Cooperative Information Systems*, pages 179–194, Irvine, CA, USA, 2002.

[10] Resource Allocation in the Grid Using Reinforcement Learning. In International Conference on Autonomous Agents archive Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004 .