Resource Allocation in Physically Distributed System using Non-Cooperative Game Theory

Bandaru Sreenivasa Rao
(211cs1056)

under the guidance of

Prof. Bibhudatta Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Resource Allocation in Physically Distributed System using Non-Cooperative Game Theory

Thesis submitted in partial fulfillment

of the requirements for the degree of

Master Of Technology

in

Computer Science and Engineering

by

Bandaru Sreenivasa Rao

(Roll 211CS1056)

under the guidance of

Prof. Bibhudatta Sahoo
NIT Rourkela

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India
May 2013

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

June 2, 2013

Certificate

This is to certify that the work in the thesis entitled **Resource Allocation in Physically Distributed System using Non-Cooperative Game Theory** by **Bandaru Sreenivasa Rao** is a record of an original work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Master of Technology* in *Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Prof. Bibhudatta Sahoo

Date:June 2, 2013
Place:NIT Rourkela

Acknowledgment

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Asst Prof. Bibhudatta Sahoo for his advice during my thesis work. As my Supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge. I am very much indebted to Prof. Ashok Kumar Turuk, Head-CSE, for his con-tenuous encouragement and support. He is always ready to help with a smile. I am also thankful to all the professors of the department for their support. I am really thankful to my all friends. My sincere thanks to everyone who has provided me with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, I am truly indebted. I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles. Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Bandaru Sreenivasa Rao

Abstract

HDCS(Heterogeneous Distributed Computing System) is a set of interconnected computing nodes so as to offers sharing of computational power, applications or network resources dynamically.Task scheduling is the crucial part which guides the resource allocation in distributed computing systems.The resource allocation in distributed computing system can be classified as centralized and decentralized.

In this thesis decentralized dynamic resource allocation scheme in distributed computing systems has been modeled as a non-cooperative game among the computing nodes and the optimal solution of this non-cooperative game is given by the nash equilibrium.The existing schemes assigns tasks to the computing nodes which are having more processing capability but in dynamic environment its not sufficient to schedule tasks based on the processing powers. Resource availability along with the processing powers of computational resources is considered to be of great significance with regard to performance issue. In the proposed scheme resource allocation is based upon the processing powers of the computing nodes, resource availability(waiting time), communication delay and the execution times of tasks at different computing nodes. This scheme minimizes makespan of the tasks while improving the fairness of the HDCS. The proposed non-cooperative scheme is giving better makespans and fairness of this proposed non-cooperative scheme is 90% i.e.,makespans at all the computing nodes in the system is almost same.

Economical models have been widely used in the distributed system for resource allocation,we have proposed an economical model proposed to minimize the resource payments based on the current states of the resources instead of cost based resource allocation. In the decentralized distributed system,we proposed an agent based cost optimization with the objective for improving the revenue of the consumers or users of the distributed system. Most importantly computational price of the tasks is minimized by paying the price based on the resource availability time(current state of the resource) instead of paying the standard prices. This

approach provides economic incentives to resource owners by fair task scheduling.If the system is unfair then the user can pay reduced price if the deadline of the tasks are not critical. Both the resource owner agent and consumer agent bargained with each other based on the current state of the resources with an objective of revising the price of the resources, which is economical to both the resource owners and consumers. All the proposed schemes has been verified through simulation against the existing game theory approaches.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Distributed Computing Systems

Distributed computing system is a set of heterogeneous computers which offers sharing of resources, those computers are connected together by some topology and do not share a memory(i.e., each computer has its own memory) or clock and execute independently. These computers communicate with each other by exchanging messages through the communication network and the resources owned and controlled by a particular computer are said to be local resources and the resources owned and controlled by other computers are said to be remote resources. Accessing remote resources are more expensive than accessing the local resources because of communication delay and network traffic. Distributed system has many advantages over a few disadvantages, those are resource sharing, availability, modularity and enhances the performance.

The performance of distributed computing system can be improved to an acceptable level simply by distributing the workload to other computers which have less workload is commonly is known as load balancing. Load distribution policies are further classified into load sharing and load balancing based on their load distribution principle. Main goal of both the policies is to reduce the workload at computers by transferring tasks to the lightly loaded computers from heavily

loaded computers. Load balancing algorithms goes one step ahead by trying to equalize the workload at all computers. Load balancing algorithms tries to equalize the workload at different computers such that every computer will be used fairly instead making of some computers are fully busy and some computing nodes are idle.

1.2 Decentralized Resource Allocation

Generally resource allocation schemes are classified as static or dynamic and centralized or decentralized.

Static approach [1–3] don't use runtime information of the system, simply they use the static information of the system while making decisions regarding task scheduling and advantage of the static approach is less overhead and easy to implement. Static resource allocation schemes assume the system information such as characteristics of tasks, workload at computing nodes and traffic in the communication network are known in earlier(constant throughout the execution)i.e.,they don't adapt to changes in the distributed environment. The main disadvantage of static algorithms is that, it assumes all the characteristics such as characteristics of tasks, computing nodes and communication network remains constant but in real time environment they may vary continuously.

On the other hand, dynamic resource allocation schemes [4–7] uses the current state of the system (i.e., uses the varying workload and network traffic during execution) while making task scheduling decisions. Even though resource allocation schemes give better performance the main disadvantage is that they are more complex than static approach because when there is any change in the system they need to collect dynamic information such as current(runtime) workload of the other computing nodes and traffic in the network.

In a centralized resource allocation schemes [5, 8] only one central computer is responsible to do the task scheduling decisions and that computer is called as the master computer i.e., all tasks are submitted at the master computer which schedules tasks to the other computers based on their processing power

or current workload or response time. Advantage of this approach is easy to implement and the main disadvantage with this approach is that if that master computer fails then entire system will go down.

For decentralized resource allocation scheme [7, 9, 10] in which each computing node makes the task scheduling decisions i.e., tasks arrive at different computing nodes with different arrival rates and each computing node schedule tasks to the other computing nodes when it is overloaded. Usually each computing node will make the decisions based on the information available from all other computing nodes in the system and this scheme is closed to individual optimal scheme means each computing node tries to improve its own objective function independent of other computing nodes.

The main disadvantage of the centralized approach is that if the master computing node fails entire system will go down which will be overcome in decentralized approach. In decentralized approach even though one computer fails remaining computing nodes will do the scheduling i.e., tasks submitted at the failed computing nodes only needs to be executed again where as in centralized approach if the central or master computing node fails we have to restart or submit all tasks again which are waiting for execution or partially executed.Disadvantage of decentralized approach is that overhead incurred in collecting the current state information of all other computers. Issam Al-Azzoni and Douglas G. Down proposed a decentralized load balancing for heterogeneous grids [10] with an objective of minimizing the communication overhead incurred in exchanging of state information.P. Neelakantan devised a adaptive load-sharing algorithm [11] which tries to balance the load by considering the connectivity among the computing nodes, processing capacity of each computing node and link capacity.

The main objective of resource allocation is for a given number of tasks find the proper allocation of tasks to other computers which optimizes the given objective function. Here objective function can be total execution time,response time,fairness,..etc.

Generally tasks arriving in a distributed system are classified into two

different classes i.e.,Multi class or Multi user.

1. Multi-User: In multi user type tasks of each user are grouped together.
2. Multi-Class: In multi-class type tasks having same arrival rate or same size are grouped together.

The optimality concept of resource allocation schemes is classified as a system optimal, class optimal and individual optimal.

1. System-optimal: In system optimal scheme all tasks are considered as single group and objective is to improve the performance of all the tasks.
2. Class-optimal: In Class optimal scheme all tasks are classified into finite number of classes based on their nature and the objective is to improve the performance of the tasks belongs to particular classes.
3. Individual-optimal: In individual optimal scheme each task tries to improve its own performance.

Computational resources are distributed and used by many users having different requirements.Users are likely to behave in a selfish manner and their behavior cannot be characterized using conventional techniques. Game theoretic models are widely used in the problems where several decision makers have to make decisions in a distributed computing systems.

1.3 Motivation

Researchers have been used game theory [12] based resource allocation schemes which provide individual optimal and system optimal solutions. Many of the past works [2, 13–16] on individual and system optimal didn't discuss about the effect of communication time,network topology and while migrating the tasks to other computers using game theory concepts.More over none of them has considered execution time of tasks at different computers i.e., they didn't consider task heterogeneity. This work concentrates on the task allocation among

computers using non-cooperative game theoretic approach by considering the ETC matrices of the tasks and communication delay for task transmission and getting back results with an objective of minimizing the makespan of tasks by improving the fairness index of the system.

1.4 Objective

The objective of thesis is to formulate a new resource allocation scheme in distributed computing system using non-cooperative game theory with the objective to minimize the makespan of the tasks by improving fairness of the system and analyse the effect of communication delay on the makespan of tasks.In the proposed scheme resource allocation will be done based on the processing powers of the computing nodes, resource availability(waiting time), communication delay and the execution times of tasks at different computing nodes.

Economical models have been widely used in the distributed system for resource allocation but objective of the economical model proposed here is to minimize the resource payments based on the current states of the resources instead of cost based resource allocation. Most importantly computational price of the tasks is minimized by paying the price based on the resource availability time(current state of the resource) instead of paying the standard prices.

1.5 Thesis Contribution

To support the thesis that the dynamic resource allocation in distributed computing systems, significant effort has delivered.Thesis contributions are

In the existing schemes tasks are assigned to the computing nodes which are having more processing capability but in dynamic environment its not sufficient to schedule tasks based on the processing powers. Resource availability long with the processing powers of computational resources is considered to be of great significance with regard to performance issue.This work has proposed a dynamic

non-cooperative approach for resource allocation which focuses on allocating tasks to the computing nodes based on the current waiting times,processing powers of computing nodes,communication delay and processing times of the tasks at different computing nodes. In any non-cooperative game when the number of players are finite then decisions of each player will have some impact on the performance of other players and optimal decisions of each player is given by the nash equilibrium.

The second contribution discussed about the agent based cost optimization while allocating the resources with an objective of getting better revenue for the user through the bargaining process and resource allocation doesn't depends on this bargaining process. There are many studies discussed about price based resource allocation in which resources are allocated to the users based on the prices but here resource allocation doesn't depends on the price per unit resource. This bargaining process use the resource allocation strategies to know the current state of the computing nodes and both the resource owner agent and consumer agent bargained with each other based on the current state of the resources with an objective of revising the price of the resources(i.e.,price per unit resource), which is economical to both the resource owners and consumers.

1.6 Organization of work

The thesis has been organist into 5 chapters

Chapter 1 Introduces the decentralized resource allocation schemes in DCS(Distributed computing system) and the problem area addressed in the thesis.

Chapter 2 presents a brief view of the proposed system model,description about some performance parameters and discuss about existing resource allocation approaches.

Chapter 3 gives some basic concepts of game theory and presents a dynamic non-cooperative approach for resource allocation.

Chapter 4 presents some existing economical models and proposed an agent based price optimization in a distributed computing system with an objective of

improving the revenue of the users.

Chapter 5 discusses the conclusion and future scope of the work.

Chapter 2

System model ,Performance metrics and Resource allocation in HDCS

2.1 Introduction

In a HDCS(Heterogeneous Distributed Computing System) tasks arriving at any computing node requires different processing times to complete because of task and system heterogeneity. Task scheduling in a distributed computing system is associated with different parameters such as task arrival time, time takes to complete the task at different computers and task migration time means time takes to transmit task and get back the results. Task scheduling is the integrated part of the distributed computing system, task scheduling is the crucial part which guides the resource allocation in distributed computing systems. The problem considered here is how to distribute or schedule tasks among the computational resources to achieve the performance goals such as minimizing the makespan and improving the fairness of the system. Task scheduling is the allocation of computational resources to satisfy the task requirements. The objective of this work is to devise a dynamic resource allocation scheme using non-cooperative game theory, which minimizes the makespan of tasks and improves the fairness of the system.

2.2 System Model

Notation	Explanation
m	- Number of computers.
M_i	- Computer i, Where 1$\leq i \leq$ m.
Pw_i	- Service rate of computer i.
λ_i	- is the task arrival rate at computer i.
λ	- is the total task arrival rate at all computers.
$Comm$	- matrix represents communication time between any two computers.
$Comm(i,j)$	- Communication time between computer i and computer j.
ETC	- Expected completion times of tasks at different computers.
T_{ij}	- is the j^{th} task at computer i where 1$\leq j \leq \lambda_j$.
$ETC(T_{ij},k)$	- Expected completion time of task j at computer k which is arrived at computer i.
$W_{time}(i)$	- is waiting time at computer i for next task.
$Ctime(T_{ij},i)$	- is completion time of j^{th} task arrived at computer i.
$Ms(i)$	- is the makespan of computer i.
Oms	- is the overall makespan of the system.
P_{ij}	- is the fraction of tasks are transfered from computer i to computer j
$SP(i)$	- Standard price of computer i.
$BP(i)$	- Bargained price of computer i.
$RP(i)$	- Revised price of computer i.

HDCS(Heterogeneous Distributed Computing System) is collection of m computing nodes and each computing node is heterogeneous in nature means each computing node has different processing capabilities. Let λ_i is the task arrival rate at computing node i and all tasks are independent.

In figure 2.1 scheduler 1 represents the scheduler of computing node 1. Arrows from scheduler 1 to other computing nodes represents the tasks are mapping from computing node 1 to other computing nodes when the computing node 1 is overloaded. Tasks arriving at any computing node can be executed by itself or it can be transferred to other computers for execution.From the above

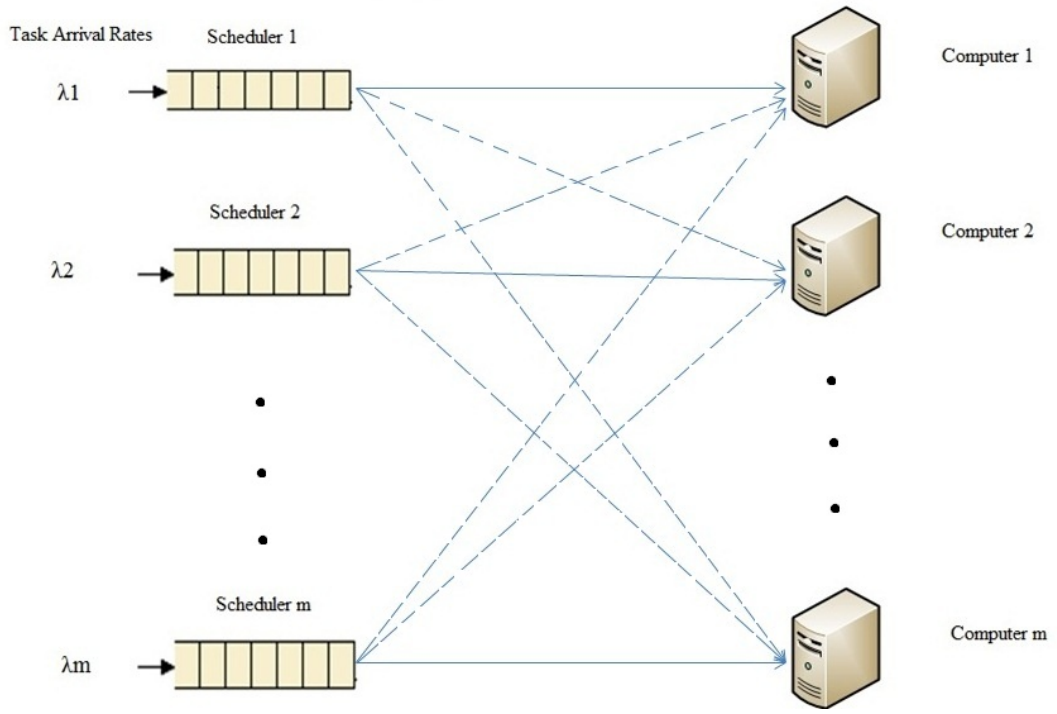

Figure 2.1: Logical representation of the decentralized distributed system

figure 2.1 dark lines represents the tasks are executed by itself and dotted lines represents tasks migrating to the other computing nodes.

HDCS(Heterogeneous Distributed Computing System) is collection of m computing nodes and each computing node in the system has its own Queue model [17, 18]. Distributed computing system is modelled as M/M/m/k/RR.
Where

M - Processing power or service rate is given by exponential distribution

M - Task Arrival rate is given by Poisson distribution

m - Number of computing nodes in the distributed system

k - Maximum number of tasks wait at any computing node

RR - Each computing node do the task scheduling in a Round robin manner
Each computing node in the system is modelled as M/M/1/k/FIFO.
Where

M - Processing power or service rate is given by exponential distribution

M - Task Arrival rate is given by Poisson distribution

1 - One computing node

k - Maximum number of tasks wait at the computing node

FIFO - Each computing node executes tasks in first in first out order

2.2.1 Task Model

Let all the tasks arriving in the system are independent, heterogeneous and they can be executed at any computing node. Tasks are independent means tasks can be executed in any order instead of waiting for some other tasks to complete. Tasks are heterogeneous in nature means tasks arriving at particular computing node requires different processing times to complete.

Here task allocation will be done using the ETC (Expected time to complete) [19] matrix which gives the expected completion of tasks at different computing nodes.

$$ETC_i = \begin{array}{c} \\ T_{i1} \\ T_{i2} \\ . \\ T_{ik} \end{array} \begin{array}{c} \begin{array}{cccc} M1 & M2 & .... & Mm \end{array} \\ \begin{pmatrix} 4 & 7 & .... & 8 \\ 6 & 8 & .... & 5 \\ . & . & .... & . \\ 9 & 4 & .... & 6 \end{pmatrix} \end{array}$$

Figure 2.2: Expected completion time of tasks at different computers

From the ETC matrix T_{i1}, T_{i2}, T_{i3},. . . . . , T_{ik} are the tasks arriving at computing node i. Here each row in the matrix specifies the expected completion times of the particular task at different computing node. Here each column in the matrix specifies the expected completion time of different tasks at particular computing node.So, here task heterogeneity is specified in columns of the matrix and system heterogeneity is specified in rows of the matrix.

The ETC matrix generated by using uniform distribution. The uniform distribution characteristics parameters a(lower bound for the range of values) and b(Upper bound for the range of values) are derived as [19]

$$\sigma = \frac{b-a}{\sqrt{12}} \tag{2.1}$$

$$\mu = \frac{b+a}{2} \tag{2.2}$$

By solving the above two equations we get

$$a = \mu - \sigma\sqrt{3} \tag{2.3}$$

$$b = 2\mu - a \tag{2.4}$$

Algorithm 1 : Algorithm for Generating ETC matrix

```
1: procedure ETCGen(PW, com, arr, m)
2:     Find minimum processing power
3:     X=Min(PW)
4:     for i = 1 : 1 : m do
5:         RPW_i = PW_i/X
6:         Relative processing powers of computing nodes
7:     end for
8:     for i = 1 : 1 : arr do
9:         C = a + (b − a) ∗ rand(1, 1)
10:        for j = 1 : 1 : m do
11:            ETC(i, j) = C/RPW_j
12:        end for
13:    end for
14: end procedure
```

ETC matrix generation of tasks at computing node i is

ETC = ETCGen (λ_i, m) where

λ_i - no of tasks arriving at computing node i

a - is the minimum expected completion time for tasks i.e., lower bound

b - is the maximum expected completion time for tasks i.e., upper bound

m - is the no of computing nodes

The above function generates expected completion times for λ_i tasks on 'm' computing nodes

Example:

An ETC matrix generation using uniform distribution

Let a=10,b=50 //Expected Completion times in between 10 and 50

λ_i=7 // Total number of tasks arriving at computing node i

m=4 //Total number of computing nodes

ETC = ETCGen (λ_i, m) $\Longrightarrow$

	M1	M2	M3	M4
T_{i1}	39	28	43	35
T_{i2}	16	27	42	22
T_{i3}	36	43	12	27
T_{i4}	31	13	26	11
T_{i5}	49	15	31	19
T_{i6}	36	17	27	38
T_{i7}	42	26	36	14

Here T_{i1} - 39 28 43 35 specifies the expected completion times of task 1 arriving at different computing nodes i.e., task 1 will 39 units of time to complete at computing node 1, 28 units of time to complete at computing node 2, 43 units of time to complete at computing node 3, 35 units of time to complete at computing node 4.

2.2.2 Task Migration Model

Tasks arriving at computing node M_i is executed at computing node M_i or it can be transferred to some other computing node M_j through the communication network. Here distributed system is considered as a decentralized system in which each computing node can schedule the tasks arriving at that computing

node to other computing nodes based on the expected completion times of a task at different computing nodes.Based on the Optimal Fraction algorithm given in chapter 3 number of tasks are transferred to each computing node will be calculated. Based on the fraction of tasks generated by OptimalFraction algorithm and ETC matrix tasks will be mapped to the other computing nodes. For example let λ_1 tasks arrived at computing node 1. Let from OptimalFractions algorithm number of tasks transferred to computing node 2 is 10 then from ETC matric 10 tasks having earliest completion times at computing node 2 will be transferred. These number of tasks transferred to particular computing nodes will decided based on the waiting times at those computing nodes.

2.2.3 Communication Model

In practical when there is less number of computing nodes we can simply connect those computing nodes using mesh topology. When the number of computing nodes grows it is very difficult maintain them in a mesh topology. So, we go for a hierarchical topology which is best suited to connect computing nodes in any organization and easy to maintain.

Physical representation of our system has shown in figure 2

From the figure 2 M_1,M_2,. . . ,M_{13} represents the computing nodes and all these computing nodes connected and communicate through the switches.

If M_1 wants to communicate with M_2 it should communicate through switch 1 i.e., $M_1 \rightarrow$ Switch 1 $\rightarrow M_2$ and hop count is 2 for M_1 and M_2. Hop is nothing but a link between any two computing nodes without any intermediate computing nodes. If computing node M_1 wants to communicate with computing node M_4 then it should go through switch 1 and switch 2 i.e., $M_1 \rightarrow$ Switch 1 $\rightarrow$ Switch 2 $\rightarrow M_4$ and hop count is 3 for M_1 and M_4. So communication time between M_1 and M_4 takes more time than M_1 and M_2 because it has to traverse one more hop. From the figure 2 the communication time between any two computing nodes is considered in terms of hop count [20] and the communication time between any two computing nodes from the above figure 2 is represented in the figure 3 in terms

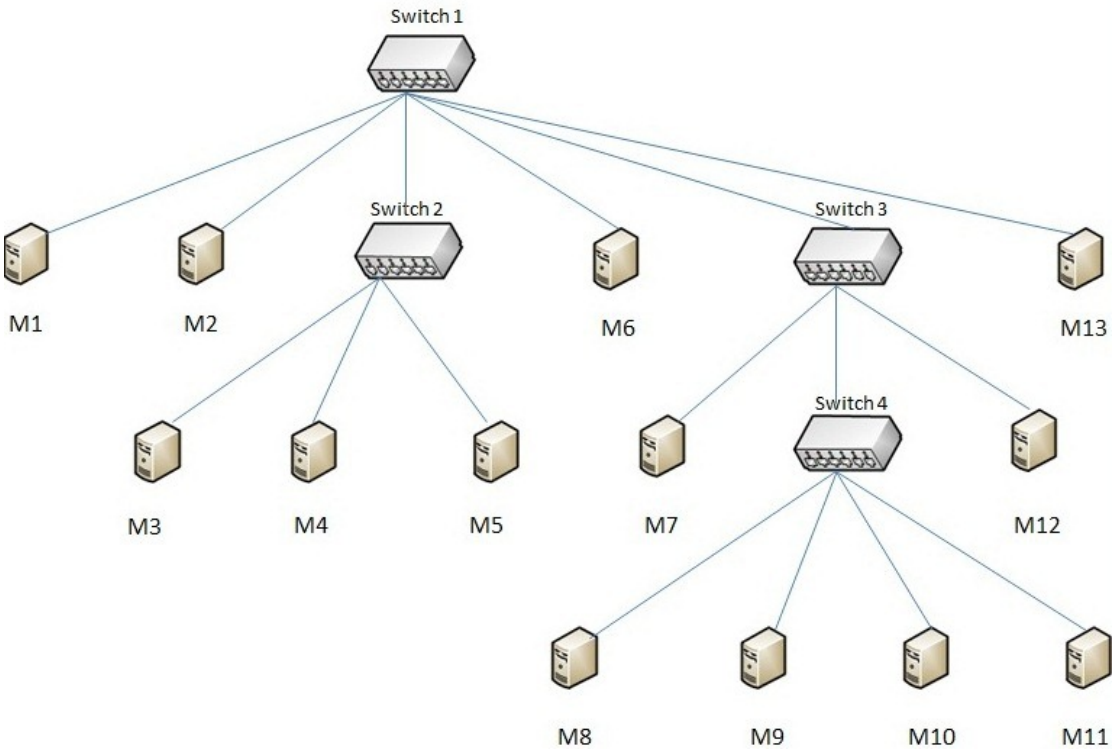

Figure 2.3: Physical representation of the Proposed HDCS

of hop count.

2.3 Performance Metrics

The performance of any HDCS is measured by different parameters which we called as performance parameters.The most commonly used performance parameters are Communication delay, Load balancing time, Scalability, Fault tolerance, Task migration cost, system utilization, response time, make span, throughput, turnaround time and fairness.

1. Communication delay: is the amount of time taken by a computer to transfer tasks to other computers [21].

2. Response Time: is the amount of time taken by the computer to respond

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
M1	0	2	3	3	3	2	3	4	4	4	4	3	2
M2	2	0	3	3	3	2	3	4	4	4	4	3	2
M3	3	3	0	2	2	3	4	5	5	5	5	4	3
M4	3	3	2	0	2	3	4	5	5	5	5	4	3
M5	3	3	2	2	0	3	4	5	5	5	5	4	3
M6	2	2	3	3	3	0	3	4	4	4	4	3	2
M7	3	3	4	4	4	3	0	3	3	3	3	2	3
M8	4	4	5	5	5	4	3	0	2	2	2	3	4
M9	4	4	5	5	5	4	3	2	0	2	2	3	4
M10	4	4	5	5	5	4	3	2	2	0	2	3	4
M11	4	4	5	5	5	4	3	2	2	2	0	3	4
M12	3	3	4	4	4	3	2	3	3	3	3	0	3
M13	2	2	3	3	3	2	3	4	4	4	4	3	0

Figure 2.4: Communication delay between computing nodes in terms of hop count

for a task [3].

$$D_j(p) = \sum_{i=1}^{m} \frac{p_{ji}}{\mu_i - \sum_{k=1}^{m} p_{jk}\lambda_k} \tag{2.5}$$

3. Load Balancing Time: is the amount of time elapses between the task arrival time and the time at which the task is finally accepted by the computer [21].

4. Scalability: is the ability of the load balancing algorithm which can perform load balancing for a distributed system with any finite number of computers [21].

5. Make Span: is the time interval between starting time of first task and maximum completion time of last task assigned to particular system [22].

6. Fault Tolerance: is the ability of a distributed system which can perform the load balancing in spite of failure in any finite number of computers [21].

7. Task Migration Cost: is the amount of time taken by a remote computer to complete execution and time takes to transfer tasks to remote computer and get back the results.

8. System Utilization: is the ratio of total task arrival rate to the total processing power of computers i.e., amount of total processing power utilized [16].

$$\rho = \frac{\lambda}{\sum_{i=1}^{m} Pw_i} \tag{2.6}$$

9. Throughput: is the total number of tasks executed per unit time.

10. Turnaround Time:is the total time taken between the submission of a task for execution and get back the complete result.

11. Fairness: is the measure of equality of completion times of tasks at different computers [16].

$$I = \frac{(\sum_{i=1}^{n} ms_i)^2}{m \sum_{i=1}^{n} {ms_i}^2} \tag{2.7}$$

12. Success Ratio: is defined as the ratio of number of tasks successfully scheduled and number of tasks arrived [23].

2.4 Resource allocation in HDCS

Game theory [12] is the formal study of decision making where several players make choices that potentially affect the interests of the other players. Where as in Non-cooperative game [24, 25] each player make decisions with out any communication such that they can improve their own pay-off. We can call this approach as selfish approach.Game theoretic models are widely used in the problems where several players have to make decisions in a distributed computing systems. If the number of decision makers are not finite then the effect of each decision maker on the outcome of the other decision makers are almost negligible. We can call this optimality as Wardrop equilibrium. If the number of decision makers are finite then the decision of each decision maker effects the outcome of the other decision makers. we can call this optimality concept as Nash equilibrium [26].

Already there exists few studies related to resource allocation problem in a distributed systems using game theory.

Daniel Grosu and Anthony chronopoulos proposed the static load balancing algorithm for a single class tasks in a distributed systems as a cooperative game [2, 13, 14] with an objective of fair task allocation such that utilization of each computer is approximately same.Here task allocation will be done based on the available processing powers of computers.

Shailendra S.Aote and M.U.Karat formulated a non-cooperative load balancing game [27] in distributed systems for minimizing response time of the tasks, here they have considered available processing powers of computers and task arrival rate as a parameters for finding the suitable computers for task allocation.

Daniel Grosu and Anthony chronopoulos proposed the selfish load balancing in a heterogeneous distributed systems as a non-cooperative game [3] to improve the response time of tasks submitted at any computer. This is a static approach which uses the processing power and task arrival rate for finding suitable computers for task assignment.

R .Subrata and Y.Zomaya proposed a load balancing problem in computational grids as a non-cooperative game by considering the processing powers of computers and transfer time of the tasks to the other computers [15] with an objective of minimizing the average completion time of the tasks.

Sagar Dhakal, Majeed M. Hayat formulated a centralized dynamic load balancing [28] in a distributed systems by considering delay incurred in a communication network and system heterogeneity in processing powers of computers. The main objective of this paper is to propose a dynamic load balancing which minimizes the overall completion times of the tasks arrived.

Preetam Ghosh, Nirmalya Roy, Sajal K. Das and Kalyan Basu proposed a game theory based pricing strategy for job allocation in mobile grids [16]. This is a static job allocation scheme with an objective of minimizing the overall response time.

Satish Penmatsa and Anthony T. Chronopoulos formulated a dynamica load balancing schemes for multi class tasks in a heterogeneous with an objective of minimizing the response time of individual computers and overall system [29, 30]. Here transmission delay is treated as a communication delay i.e., ratio of task size and bandwidth and they didnt consider any particular topology.

Satish Penmatsa and Anthony T. Chronopoulos proposed price based job allocation schemes for computational grids [31]. Here prices charged by grid owners are calculated based on the pricing model using a bargaining game theory and these prices are then used for job allocation to different computers with an

objective of minimizing cost for grid users. One scheme they tried to minimize the cost of the entire grid system and second scheme tries to minimize the cost for each server. Xin Bai and Dan C.Marinescu discussed about some pricing models [32] task allocation in distributed system and analyzed the performance between utilization and the price paid for resources.

An agent-based methodology is developed for building large-scale distributed systems with highly dynamic behaviors [33].A combination of intelligent agents and multi-agent approaches is applied to both local grid resource scheduling and global grid load balancing [34] and [35] applied use of economic agent in grid computing. Chunlin and Layuan presented a Multi-economic agent model [35] for grid resource management and the system model is described that allows agents representing various grid resources and grid users to interact without assuming priori cooperation.The grid task agents buy resources to complete tasks and grid resource agents charge the task agents for the amount of resource capacity allocated. This paper provides a price-directed market-based algorithm for solving the grid task agent resource allocation problem.

2.5 Conclusion

This chapter introduced a HDCS(Heterogeneous Distributed Computing System) model which helps in understanding the relationship between different components of the proposed model and presented a different performance metrics of the distributed computing system. Existing studies of game theory based resource allocation are presented here and none of these studies specified in the related work considered about execution times of tasks at different computing nodes when the tasks are migrating i.e.,they didn't consider task heterogeneity and few papers discussed about task transfer delay but no one discussed about communication time based on the network topology [36]. So,the main objective of this thesis is to propose a non-cooperative game theoretic approach for resource allocation in distributed computing systems by considering task,system heterogeneity and communication delay based on the network topology. Some

economical resource allocation approaches are presented here and in this approaches task scheduling is based on the pricing policies(i.e.,price per unit resource)defined by the resource owners.

Chapter 3

Decentralized Resource Allocation Scheme in HDCS

3.1 Introduction

In a decentralized distributed computing system each computing node can make decisions regarding scheduling the tasks which arrive at different computing nodes.Task allocation in decentralized computing system is a multiple decision maker activity and problem of decision making can be modelled using game theory, which is widely used in the problems involving several independent decision makers. The decision making of task scheduling has been modelled as the non-cooperative game and the optimality concept of this non-cooperative game is represented by the nash equilibrium [26]. In this dynamic non-cooperative approach communication delay between any two computing nodes is taken as hop count. This approach focuses on allocating tasks to the computing nodes based on the current waiting times,processing powers of computing nodes,communication delay and processing times of the tasks at different computing nodes. If there are m computing nodes then instead of assigning tasks to each computer, this approach schedules tasks to the selected computing nodes who available earlier. These selected computers are identified by the OptimalFraction algorithm. The performance of this non-cooperative scheme is analysed under the performance

parameters makespan,fairness and communication delay.

3.2 Non-cooperative game theory in distributed decision making

Game theory [12] is a formal way to analyse the interaction among rational players. In any game decision makers are called as the players. If there is a single player then game becomes a decision problem. An interaction among players is anything that affects the decisions of at least one other player. Otherwise the game is simple a series of independent decision problems.
A strategic form of a game is represented as follows:

1. Number of players: m computers
2. Strategy: Strategy set of player i, p_i is the set of actions taken by the player i.
3. Preferences: Each player i,gives preference to the strategy profile p to p' if $U(p') \geq U(p)$ then p' is giving better profit.

The two main classes of the games are cooperative and non-cooperative.

In cooperative game [2, 13, 14] several decision makers cooperate in making the decision such that each player will operate at the optimum.

In a non-cooperative game [24, 25] every player will try to improve their own objective function independent of other players and they all eventually reach an equilibrium.

In a non-cooperative game when the number of players are finite then decisions of each player will have some impact on the performance of other players and optimal decisions of each player is given by the nash equilibrium [26]. Nash equilibrium of a m-player finite game is defined as a strategy profile $p = \{p_1, p_2, ..., p_m\}$, where $p_1, p_2, ..., p_m$ are the strategies of the players. At nash equilibrium no player improve its own profit unilaterally by changing from its

current strategy to another feasible strategy i.e., at the equilibrium point no player will get benefit by changing his strategy to another strategy while the strategies of the other players are fixed or constant. To reach nash equilibrium each player i computes the best strategy p_i i.e., strategy which gives the maximum pay-off for the player i and this strategy is calculated based on the decisions of other players. Each player calculates its own strategy continuously until an equilibrium is reached and once an equilibrium reached no player will change to another strategy until there is some change in the strategies of the other players.

Example : *Finding Nash Equilibrium Point*

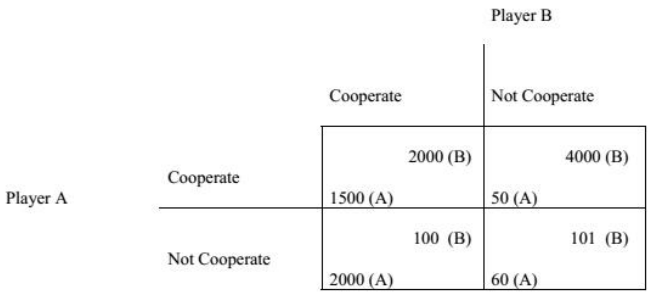

Figure 3.1: Payoff matrix for two players

Let us take 2 player game,from the figure 3.1 each player can make two decisions and there is profit associated with those decisions against the decisions of the other player.

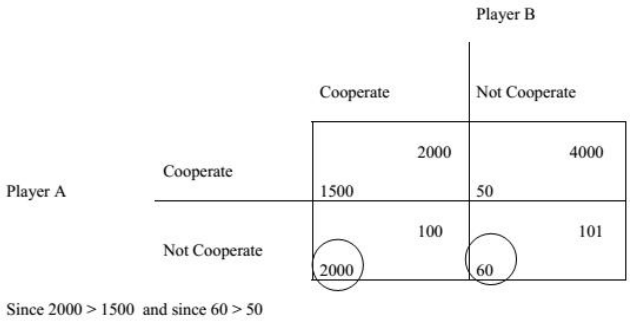

Figure 3.2: Player A best response to all of player B actions

In the figure 3.2 player A has two choices weather to cooperate or not cooperate and there is price associated with those choices. If player A wants to

cooperate then he gets profit 1500 or 50 based on the player B decisions and if player A don't want to cooperate then he gets profit 2000 or 60 based on the player B decisions . Here player A decides not to cooperate because he is getting more profit by not cooperating as compared with cooperating against player B decisions i.e.,$2000 \geq 1500$ and $60 \geq 50$. So, player A prefers to not cooperate with player B.

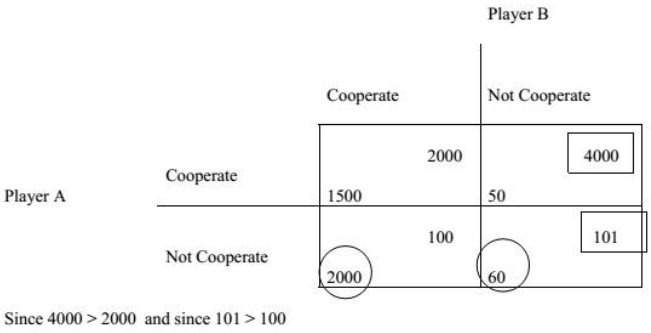

Figure 3.3: Player B best response to all of player A actions

From the figure 3.3 player B has two choices weather to cooperate or not cooperate and there is price associated with those choices. If player B wants to cooperate then he gets profit 2000 or 100 based on the player A decisions and if player B don't want to cooperate then he gets profit 4000 or 101 based on the player A decisions . Here player B decides not to cooperate because player B is getting more profit by not cooperating as compared with cooperating against player A decisions i.e.,$4000 \geq 2000$ and $101 \geq 100$. So, player B prefers to not cooperate with player A.

Figure 3.4 has the intersection point of the actions of both players is called the nash equilibrium, which is point at which both the player will get better profit. At this nash equilibrium point no player gets better profit by changing their decisions while the decisions of the other players are fixed or constant. From the figure 3.4 nash equilibrium point is (60,101), where the decision of player A is not-cooperate and the decision of player B also not-cooperate. If player A changes to another decision cooperate and player B decision is fixed means not-cooperate then profit received by the player A is 1500 which is less compared

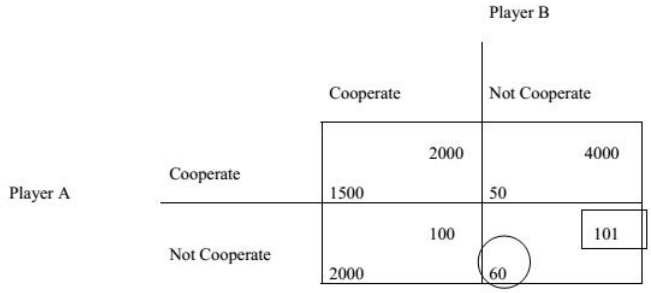

Figure 3.4: A Nash equilibrium exists where Player B best response is the same as Player A best response

with the earlier profit 2000. So, at the nash equilibrium point no player will get benefit by changing their decisions unilaterally. In the proposed non-cooperative scheme with m computing nodes, optimal decisions regarding task scheduling of this m-player game is given by the nash equilibrium.

3.3 Non-Cooperative Resource Allocation approach

Let there are m computing nodes in a HDCS(Heterogeneous Distributed Computing System) and each computing node make decisions independently with out any cooperation among them i.e., each computer will make decisions selfishly to improve their own makespan.This type of game is called as non-cooperative game and equilibrium point of this game is given by nash equilibrium.

At Nash equilibrium, no player can improve its own performance unilaterally by changing from its current strategy to another feasible strategy i.e., at the equilibrium point no player will get benefit by changing his strategy to another strategy while the strategies of the other players are fixed or constant. In a strategy profile each computing node will find the best strategy which improves makespan of its own tasks based on the strategies of other computing nodes.

$$p_j \, \epsilon \, min \, MS_j(p_1, ...\overline{p_j}, ...p_m) \tag{3.1}$$

The makespan of computing node j is given by

$$Ms(j) = max[Ctime(1), Ctime(2), ...Ctime(\lambda_j)] \tag{3.2}$$

The completion times of tasks arrived at computing node j is given by

$$Ctime(p_j) = \sum_{x=1}^{m}[\sum_{l=1}^{p_{jx}}(T_{jl} + Comm(j,x)) + Wtime(x)] \tag{3.3}$$

Where

p_{jx} - is the number of tasks transfered from computing node j to x.

T_{jl} - is the l^{th} task which is arrived at computing node j.

$Wtime(x)$ - is the waiting time at computing node x.

$Comm(j,x)$ -is the communication time between computing node j and x.

The overall makespan of the HDCS is given by

$$Oms = max[Ms(1), Ms(2), ...Ms(m)] \tag{3.4}$$

The strategy profile of a game is given by $p = \{p_1, p_2, ..., p_m\}$ which is the set of strategies of all players in the game and $p_j = (p_{j1}, p_{j2}, ...p_{jm})$ which is called as strategy of computing node j i.e., in a non-cooperative game the strategy of computing node j is defined as a set of actions taken by the computing node j. Here p_{ji} is the fraction of tasks computing node j maps to the computing node i which is calculated based on the current state of other computing nodes.

The objective of computing node j is to minimize the makespan of the tasks arrived at computing node j.While transferring tasks to other computing nodes, each computing node should satisfy the following conditions

1. Positivity: $p_{ji} \geq 0, i = 1, 2, ..., m, j = 1, 2, ..., m$ means computing node j can send some fraction of tasks or don't send tasks to computing node i.

2. Conservation: $\sum_{i=1}^{m} p_{ji} = 1, j = 1, 2, ...m$ means the number of tasks arrived at computing node j is equal to the total number of tasks transmitted to the other computing nodes and fraction of tasks executed by a computing node j.

To reach nash equilibrium each computing node will find a best strategy p_i i.e., strategy which gives the minimum makespan for computing node i and this strategy is calculated based on the current state of the other computing nodes. Each computing node calculates its own strategy continuously until an equilibrium is reached and once an equilibrium reached no computing node will change to another strategy until there is some change in the strategies of the other computing nodes.

The strategy p_j of computer $j(j = 1, 2, ..., m)$ is calculated by using the OptimalFraction algorithm given below.p_{ij} is the number of tasks needs to map from computing node j to computing node i is calculated based on the equation given below

The strategies of each computing node can be calculated in two ways

1. Here current states of the computing nodes are considered in terms of their waiting times i.e., OptimalFraction algorithm is function of only waiting times and it didn't consider their processing capabilities.

$$p_{ij} = \begin{cases} \frac{\frac{1}{k}\sum_{r=1}^{k} W_{time}(r) - W_{time}(j)}{\sum_{l=1}^{k}\{\frac{1}{k}\sum_{r=1}^{k} W_{time}(r) - W_{time}(l)\}} & if 1 \leq j \leq k \\ 0 & if k \leq j \leq m \end{cases} \tag{3.5}$$

   Where k is the minimum index which satisfies the condition

$$W_{time}(k) \leq \frac{1}{k}\sum_{i=1}^{k} W_{time}(i) \tag{3.6}$$

2. In second approach current states of the computing nodes are taken as the ratio of processing powers of the computing nodes and their current waiting times i.e., OptimalFraction algorithm is a function of both the waiting times of the computing nodes and their processing capabilities.

$$p_{ij} = \begin{cases} \frac{Pw_j / W_{time}(j)}{\sum_{l=1}^{k} Pw_l / W_{time}(l)} & if k \leq j \leq m \\ 0 & if 1 \leq j \leq k \end{cases} \tag{3.7}$$

   Where k is the minimum index which satisfies the condition

$$\sqrt{\frac{pw_k}{W_{time}(k)}} \leq \frac{\sum_{i=1}^{k} \frac{pw_i}{W_{time}(i)}}{\sum_{i=1}^{k} \sqrt{\frac{pw_i}{W_{time}(i)}}} \tag{3.8}$$

Algorithm 2 : Algorithm for finding OptimalFraction of tasks for Allocation

1: **procedure** OPTIMALFRACTION(W_{time}, Pw)
2: Sort all computing nodes based on the ratio of processing power to waiting times $\frac{Pw_1}{W_{time}(1)} \geq \frac{Pw_2}{W_{time}(2)}, \ldots\ldots \frac{Pw_m}{W_{time}(m)}$
3: $k = m$;
4: $t = \frac{\sum_{i=1}^{k} \frac{pw_i}{W_{time}(i)}}{\sum_{i=1}^{k} \sqrt{\frac{pw_i}{W_{time}(i)}}}$
5: **while** $t > \frac{Pw_k}{W_{time}(k)}$ **do**
6: $p(k) = 0$
7: $k = k - 1$
8: $t = \frac{\sum_{i=1}^{k} \frac{pw_i}{W_{time}(i)}}{\sum_{i=1}^{k} \sqrt{\frac{pw_i}{W_{time}(i)}}}$
9: **end while**
10: **for** $j = k$ to m **do**
11: $p(j) = \frac{Pw_j / W_{time}(j)}{\sum_{i=k}^{m} Pw_i / W_{time}(i)}$
12: **end for**
13: **end procedure**

Example 1: *Finding optimal fractions based on the waiting times*

Let there are 6 computing nodes and 10 tasks arrived at computing node 1

Processing powers of different computing nodes are Pw=[10 20 30 10 50 100]

Waiting time at different computing nodes are W_{time}=[100 120 50 120 60 70]

W_{avg}=86

Send 0% tasks to the computing nodes having more waiting times than average waiting time.

p = [0 0 - 0 - -]

t1=86-50=36,

t2=86-60=26,

t3=86-70=16

t4=36+26+16=78

p = [0 0 $\frac{t1}{t4}$ 0 $\frac{t2}{t4}$ $\frac{t3}{t4}$]

p = [0 0 $\frac{36}{78}$ 0 $\frac{26}{78}$ $\frac{16}{78}$]

p = [0 0 50 0 30 20]

Means sending no tasks to the computing nodes 1,2,4 and 50% of the tasks to computing node 3, 30% of the tasks to computing node 5,20% of the tasks to computing node 6

In the example 1 OptimalFraction of tasks is calculated based on the current waiting times.According to this example computing nodes having more waiting times are getting less percentage of tasks and the computing nodes having less waiting times are getting more tasks with an objective of equalizing the workload at all the computing nodes. From the above example 1 computing node 3 having processing power 30 is getting 50% of tasks and computing node 6 having processing power 100 is getting 20% of tasks. In this example computing nodes having less processing power are getting more tasks than the computing nodes having more processing powers.So, while assigning tasks we should consider both the waiting times at the computing nodes and their processing powers.

Example 2: *Finding optimal fractions based on both the processing powers and waiting times*

Let there are 6 computing nodes and 10 tasks arrived at computing node 1

Processing powers of different computing nodes are Pw=[10 20 30 10 50 100]

Waiting time at different computing nodes are W_{time}=[100 120 50 120 60 70]

let r is the ratio of processing powers of computing nodes to the waiting times at those computing nodes.

r=[0.10 0.16 0.60 0.08 0.83 1.42]

here t will be calculated from the OptimalFraction algorithm, $t = 0.34$ means here tasks are assigned to the computing nodes having the ratio of processing power to waiting time is greater than the t

Send 0% tasks to the computers if $r_k \leq t$

p = [0 0 - 0 - -]

sum=0.60+0.83+1.42=2.85

p = [0 0 $\frac{0.60}{2.85}$ 0 $\frac{0.83}{2.85}$ $\frac{1.42}{2.85}$]

p = [0 0 20 0 30 50]

Means sending no tasks to the computing nodes 1,2,4 and 20% of the tasks to computing node 3, 30% of the tasks to computing node 5,50% of the tasks to computing node 6

In example 2 optimal fraction of tasks is calculated based on the current waiting times and processing powers of the computing nodes. From this example computing nodes having highest ratio of processing powers to waiting times will get more percentage of tasks. Here computing node 3 is getting less tasks than 6 even though it has less waiting time because of high ratio of processing power to the waiting times. Even though computing node 6 has more waiting time than computing node 3 it can finish tasks earlier than computing node 3 because of its high processing capability.

3.4 Non-cooperative algorithm for Resource allocation

Based on the load fractions calculated by the OptimalFraction algorithm to each computing nodes,in this proposed decentralized resource allocation approach each computer will do the task scheduling decisions based on the ETC matrix and updates their strategies time to time by considering the current workloads of the computing nodes in a round robin manner.

Each computer will update their strategies until they reach nash

equilibrium. Once they reach nash equilibrium then they will use the same strategies to make scheduling decisions because at nash equilibrium no computer will get any benefit by changing their strategies to other feasible strategies while the strategies of the other computers are fixed. Each computer updates their strategies from time to time by finding the optimal strategies.For finding nash equilibrium there should be some communication among the computers.

In this non-cooperative resource algorithm each computer updates their strategies in a round robin fashion.This algorithm executes periodically when there is any change in the system parameters. To reach nash equilibrium each computer calculates its optimal strategies based on the current state of other computers in a round robin fashion. In a decentralized HDCS(Heterogeneous Distributed Computing System) each computer has its own agent(scheduler process) which makes resource allocation decisions and communicates with the scheduler agents of other computers for information exchange of the current state.These agents are responsible for estimating tasks arrival rate at particular period of time and these agents receive the current state of other computers and calculates strategies by using the OptimalFraction algorithm based on the current state of the computers. Once strategy of p_j is calculated then agent will do the mapping of tasks to the other computers based on the p_{ji}.

The execution of the algorithm restarts periodically or when it reaches to the predefined threshold value. Let the threshold value is 0.8, if the fairness index of the system is less than 0.8 then stop the algorithm and restart again. If the fairness index is 1 then makespan at each computer is approximately same. If the fairness index is 0.8 then for 80% of computers makespan is approximately equal.

After finding the fraction of tasks to each computer by computer i then computer i uses ETC matrix to map which task to which computer. For example let λ_1 tasks arrived at computer 1. From OptimalFraction algorithm number of tasks transfered to computer 2 is 5 then from the ETC matrix 5 tasks having earliest completion times at computer 2 will be transferred to computer 2. These number of tasks transferred to particular computer will decided based on the

processing power of computing nodes and waiting times at those computing nodes.

Example 3: *Allocating tasks based on the waiting times* In this example fraction of tasks to each computer is calculated using the equation 3.5 based on the waiting time at different computers.

Processing powers Pw=[10 20 30 10 50 100]

Waiting times Wt=[100 120 50 120 60 70]

ETC Matrix

Tasks	M1	M2	M3	M4	M5	M6
T1	89	44	30	89	18	9
T2	80	40	27	80	16	8
T3	100	50	33	100	20	10
T4	68	34	23	68	14	7
T5	72	36	24	72	14	7
T6	66	33	22	66	13	7
T7	78	39	26	78	16	8
T8	90	45	30	90	18	9
T9	124	62	41	120	24	12
T10	99	49	33	99	20	10

Figure 3.5: Allocating Tasks based on the waiting times

From the figure 3.5 processing powers of computing nodes are pw=[10 20 30 10 50 100] ,waiting times at those computing nodes are wt=[100 120 50 120 60 70] and expected completion times of the tasks at different computing nodes are given by the ETC matrix in figure 3.5.

In figure 3.6 p is the number of tasks are mapping to each computing node is calculated based on the waiting times as shown in the example 1 using OptimalFraction algorithm. Here we are not mapping any tasks to the computers 1,2 and 4 because of having more waiting times and we are sending 5 tasks to the computing node 3 which are having less execution times at that computing node i.e., tasks $T2, T4, T5, T6$ and $T7$ have a less execution times at computing node 3 as compared with the other tasks.

From the figure 3.7 number of tasks need to map to a computing node 5 are 3. So,tasks $T1, T3 and T8$ are selected to send because of having less execution times.

Algorithm 3 : Non-cooperative Resource Allocation

1: **procedure** NON-COOPERATIVE($Pw_1, Pw_2, ..., Pw_m$)
2: **for** each computer i(i = 1,2,...m) **do**
3: ETC = ETCGen(λ_i, m)
4: **if** ($iteration == 1$) **then**
5: p=Proportional(Pw, λ_i);
6: **else**
7: p=OptimalFraction($Wtime, Pw, \lambda_i$);
8: **end if**
9: **for** each computer j(j=1,2,...m) **do**
10: Map P[j] tasks having less execution time at computing node j
11: $Wtime(j) = UpdateWaitingtimesatj$
12: $Ctime(i,:) = FindCompletiontimesofP[j]tasks$
13: **end for**
14: **end for**
15: **for** each computer i(i=1,2,...m) **do**
16: Ms(i)=Find MakeSpan at Computer i
17: **end for**
18: Oms=ComputeOverallMakeSpan
19: $I = Calculatefairness$
20: **if** ($I \leq threshold$) **then**
21: break
22: **else**
23: Goto Step2
24: **end if**
25: **end procedure**

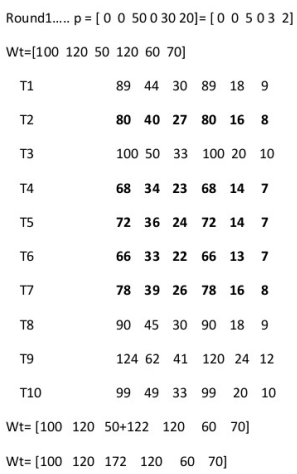

Figure 3.6: Allocating 5 tasks to computing node 3

From the figure 3.8 number of tasks need to map to a computing node 6 are 2. So,tasks $T9, T10$ are selected to send because of having less execution times. After this allocation new waiting times at different computers are wt=[100 120 172 120 116 92]. Fairness [16] of the system is given by

$$I = \frac{(\sum_{i=1}^{n} wt_i)^2}{m \sum_{i=1}^{n} {wt_i}^2} \tag{3.9}$$

form the above equation I is calculated as 0.94 means 94% of the computing nodes have approximately same completion times.

Example 3:*Allocating tasks based on the processing power of computing nodes and their waiting times*

In this example number of tasks to each computing node is calculated based on the ratio of processing power to waiting times. Here computing node having highest ratio will receive high number of tasks for execution and the strategy vector p will be calculated as shown in example 2 using the OptimalFraction algorithm. Here p is given by

$$p_{ij} = \begin{cases} \frac{Pw_j/W_{time}(j)}{\sum_{i=1}^{k} Pw_i/W_{time}(i)} & if k \leq i \leq m \\ 0 & if 1 \leq i \leq k \end{cases} \tag{3.10}$$

After finding the vector p task allocation will be done From the figure 3.9 we are

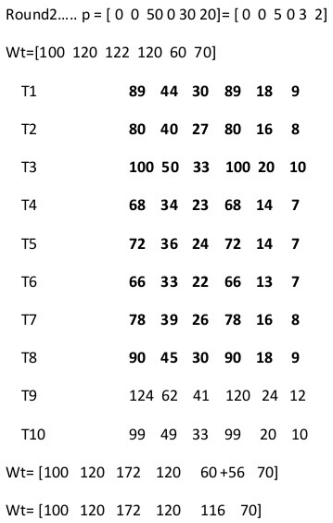

Figure 3.7: Allocating 3 tasks to computing node 5

not mapping any tasks to the computers 1,2 and 4 because of having more waiting times and we are sending 2 tasks to the computing node 3 which are having less execution times at that computing node i.e., tasks $T4$ and $T6$ will be transfered to computing node 3 because these have a less execution times at computing node 3 as compared with the other tasks .

From the figure 3.10 number of tasks need to map to a computing node 5 are 3. So,tasks $T2, T5$ and $T7$ are selected to send because of having less execution times. Even though tasks $t4, T6$ have less execution times at computer 5, the reason for selecting other tasks is those tasks are already assigned to computer 3.

From the figure 3.11 number of tasks need to map to a computing node 6 are 5 because of having the high ratio of processing power to the waiting times and the tasks $T1, T3, T8, T9$ and $T10$ are selected to transfer. After this allocation new waiting times at different computers are wt=[100 120 95 120 106 120]. Fairness [16] of the system for this allocation is given by

$$I = \frac{(\sum_{i=1}^{n} wt_i)^2}{m \sum_{i=1}^{n} {wt_i}^2} \tag{3.11}$$

from the above equation fairness is calculated as 0.98 means 98% of the computing nodes have approximately same completion times.

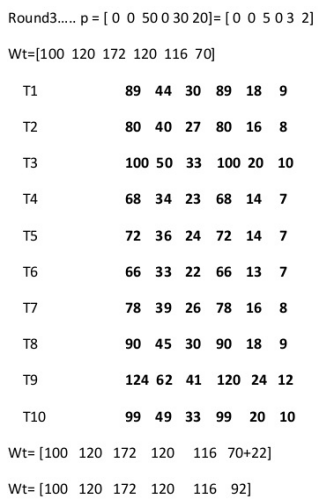

Figure 3.8: Allocating 2 tasks to computing node 6

Based on the example 3 and example 4 we can conclude by considering the ratio of processing powers of computing nodes to their waiting times instead of considering waiting times only, we are getting better makespans and fairness.

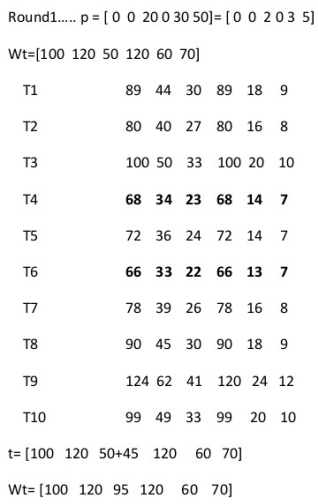

Figure 3.9: Allocating 2 tasks to computing node 3

3.5 Simulation and Results

Simulation of dynamic non-cooperative resource allocation scheme is carried out by the simulator designed using matlab R2010a and the performance of the system is analysed under different performance metrics makespan,fairness and the effect of communication delay on makespan of the system.

Simulation Parameters:

1. *Simulation tool:* Matlab R2010a
2. *Number of computers :* 10-100
3. *Number of tasks:* 100 - 1000
4. *Task arrival :* Poisson distribution
5. *ETC :* Uniform distribution
6. *Communication Time:* In terms of Hop count
7. *Performance metrics:* Makespan, fairness
8. *Heuristics considered:* Overall optimal scheme scheme,Non-cooperative approach

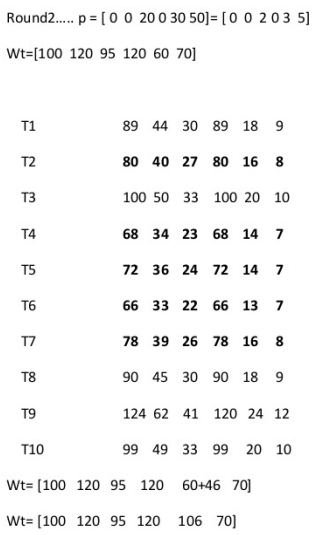

Figure 3.10: Allocating 3 tasks to computing node 5

9. *Objective:* Minimize the makespan of tasks,improve the fairness of the distributed computing system and analyse the effect of communication delay on makespan.

Graphs shown in the figure 3.12 depicts effect of workload on makespan and it compares the makespan of proposed non-cooperative with the OPTIM [37] scheme at different work loads. From the figure 3.12 proposed non-cooperative scheme gives better makespan as compared with the OPTIM [37] scheme.Here OPTIM scheme schedules the tasks according to the processing power of the computing nodes and tasks arrival rate but it does not consider the current waiting times at different computing nodes i.e., it always assigns same number of tasks to each computing node based on the processing powers, where as in the proposed non-cooperative scheme scheduling will be done based on the waiting times at different computing nodes i.e., more tasks will be transferred to computing nodes having less waiting times.

Results of this proposed non-cooperative scheme is compared with the OPTIM .

Overall Optimal Scheme [37]: This scheme calculates the load fractions based on the average processing powers of computing nodes and the task arrival rate.This allocation scheme may not minimize the makespan of the system and this

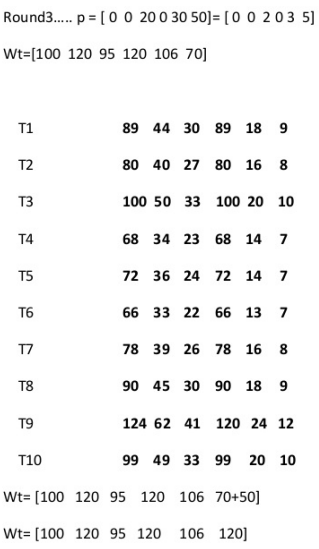

Figure 3.11: Allocating 5 tasks to computing node 6

is a static approach which don't consider the waiting times and communication times between the computational nodes.

Proportional Scheme [38]: According to the proportional scheme each computing node allocates tasks to other computing nodes based on the proportion to their processing power i.e., computing node having high processing power will get more number of tasks than the computing node having low processing power.

$$p_i = \lambda * \frac{Pw_i}{\sum_{i=1}^{m} Pw_i} \tag{3.12}$$

The following table gives the makespan of tasks with different arrival rates from 100 to 1000. It depicts the makespan for both proposed non-cooperative and OPTIM approaches which is given in the figure 3.12.

Fairness index I is used to measure the fairness [16] of load balancing schemes.Fairness index is given by the equation

$$I = \frac{(\sum_{i=1}^{n} ms_i)^2}{m \sum_{i=1}^{n} {ms_i}^2} \tag{3.13}$$

Here ms is the vector $ms = (ms_1, ms_2, ..., ms_m)$ is the makespan of the tasks at each computer. Here fairness index measures the equality of expected

Number of Tasks	Non-cooperative	OPTIM
100	653	1250
200	1120	2235
300	2800	4342
400	4487	4969
500	4785	6175
600	4896	8253
700	5125	8268
800	4075	9951
900	7385	11179
1000	9527	12195

Table 3.1: MakeSpan of Non-coop vs OPTIM

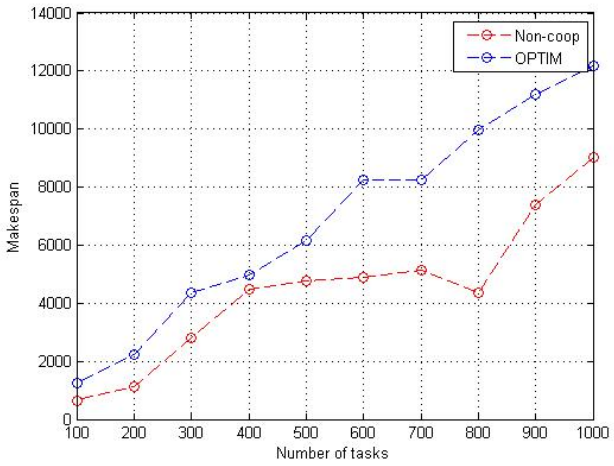

Figure 3.12: Makespan vs Number of Tasks

completion times of tasks at different computers. If all computers have same expected completion times the I=1, means system is 100% fair to all the computers and completely load balanced i.e., here I gives percentage of computers having approximate completion times. If difference between ms_j of different computers increases then load balancing scheme will favors only for some computers. The fairness index for varying number of computers are shown in the figure **??**.

Figure 3.14 shows the fairness index of the system.Fairness of the proposed non-cooperative scheme with waiting times only and the ratio of processing power to the waiting times are compared here.

Figure 3.15 shows the effect of communication delay on makespan of the tasks arrived in the system. In this non-cooperative approach all computers connected through a hierarchical topology and the communication delay between

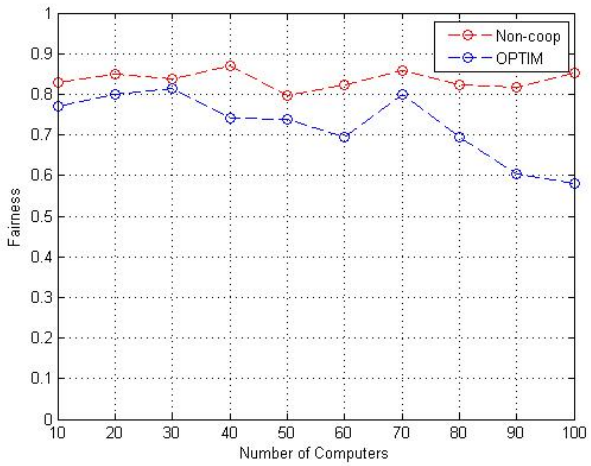

Figure 3.13: Fairness vs Number of Computers

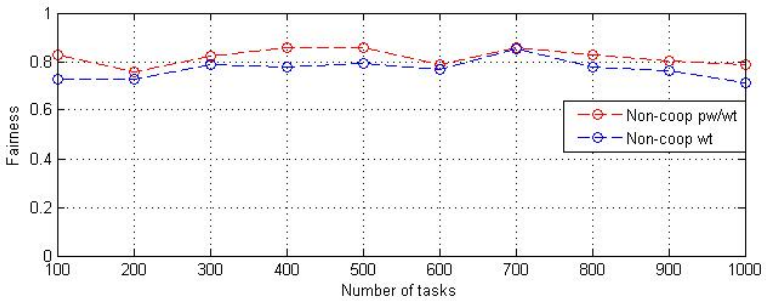

Figure 3.14: Fairness of non-cooperative schemes

any two computers is given by hop count between those computers. Hop is defined as a link between any two computers without intermediate computers or switches and hop count is the number of hops between the source and destination computers. In figure 4.4 both the graphs are generated using the proposed non-cooperative approach but in one graph communication delay for task transmission is also considered in terms of hop count.

The following table gives the makespan of tasks with different arrival rates from 100 to 1000. It depicts the makespan for both proposed non-cooperative and proposed non-cooperative with communication delay which is given in the figure 3.15.

From the graphs 34% of delay is caused due to communication delay

Number of Tasks	Non-cooperative	Non-coop with Comm delay	Delay due to comm
100	549	590	41
200	1044	1383	339
300	1126	1752	626
400	2195	2294	99
500	2932	3604	672
600	3449	4166	717
700	3915	5435	1520
800	4319	6183	1864
900	4680	6991	2311
1000	4934	6659	1725

Table 3.2: MakeSpan of Non-coop vs Non-coop with communication delay

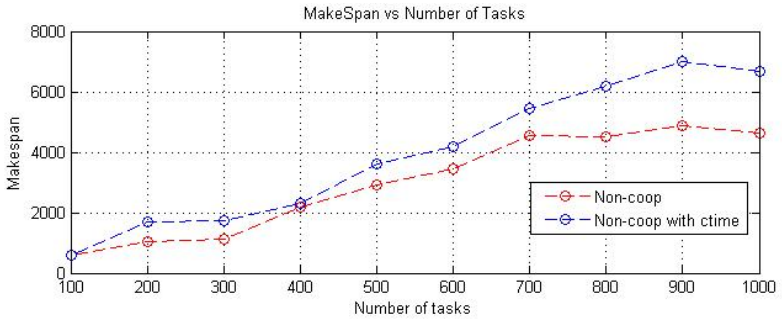

Figure 3.15: MakeSpan vs Number of tasks with communication delay

In hierarchical topology all computers connected through switches and hop count between computers will vary based on the number of switches through which they are connected together. From the figure 3.16 we can say that makespan of tasks is also depends on the number of switches through which all computers are connected. In this simulation number of tasks arriving are fixed and repeated the experiment for different number of switches varying from 10 to 50. From the figure 3.16 we get better makespan at 20,50 as compared with the 20,30 because from figure 2.3 if computer M1 needs to send more tasks to computers under switch 4 which needs to traverse more number of hops. Clearly it will increase the hop count which is taken as communication delay and then it causes to increase in the makespan of tasks because of increasing communication delay. Here we're getting minimum makespan even at 40,50 number of switches because from figure 2.3 if computer M1 needs to send more tasks to the computers under switch 1 and switch 2 which needs to traverse less number of hops which in turn minimizes

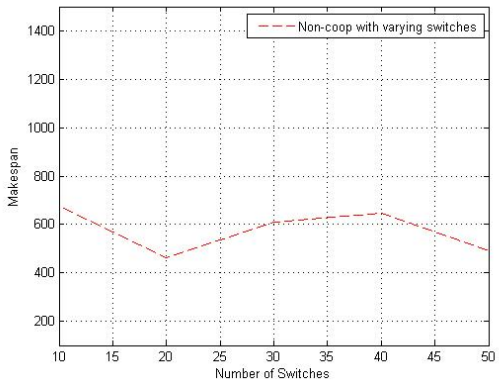

Figure 3.16: Effect of Number of switches on Makespan

the communication delay i.e., communication delay is taken in terms of hop count. From the figures 3.16 and 3.17 we can conclude the makespan will greatly effect by

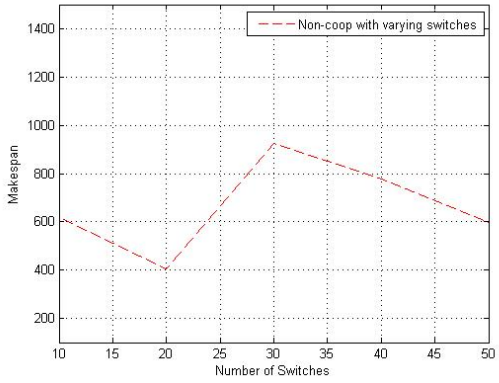

Figure 3.17: Effect of Number of switches on Makespan

number of switches and to which computers tasks are transmitted. Clearly from the figure 3.17 more tasks send to the computers under distant switches instead of computers under near by switches.

Figure 3.18 depicts the makespan when the number of computing nodes are increasing, clearly we can say that here we are getting better makespans when the number of computational nodes are increased.

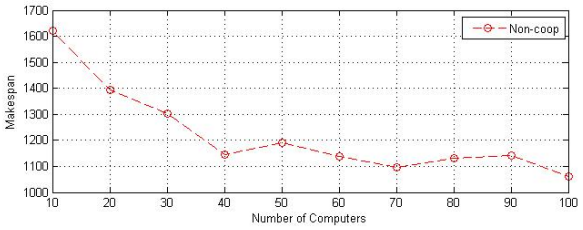

Figure 3.18: Makespan vs Number of computing nodes

3.6 Conclusion

This chapter presents some basic concepts of game theory and implementation of proposed non-cooperative resource allocation in distributed computing systems. The performance of this approach is analysed by the performance parameters makespan, fairness and communication delay. From the above results we can conclude that the proposed non-cooperative scheme is giving better makespans and fairness of this proposed non-cooperative scheme is 90% i.e.,utilization of all the computing nodes in the system is almost same. 30% of the makespan is due to the communication delay in the network and the communication delay highly depends on the network topology.

Chapter 4

Agent based Cost optimization in Resource allocation

4.1 Introduction

Autonomous agents are intelligent entities that can operate on behalf of the users autonomously to solve the problems, negotiate with other agents, learn from the past and predict upcoming events. Agent are used in various application domain such as industrial applications viz. process control, commercial applications viz. electronic commerce, business process management, medical applications viz. patient monitoring ,health care.

Agents get the problem from the users or other agents, discover needed resources, consult with other agents (negotiation) and offer a proper solution. They also learn from the past, update their knowledge and predict the future events [39]. Agent technology is new in distributed system and can be used in computing network. Agents may be autonomous and intelligent entities (i.e. software) on a network that reside on the nodes or may travel between them. They get the problem from the users or other agents, discover needed resources, consult with other agents and offer proper solution. The main difference between agent and scheduler is coordination, cooperation and learning [39]. Agents work together, use the resources located on each other optimally and work as a team to

solve a problem. Agents are flexible entities and are capable to adapt themselves to new environments. This means agents are well suited in dynamic environment. Even though agents are independent they always communicate with others to discover needed resources.

Following are some properties of the agent

- Autonomous - agents are proactive, goal directed and is capable of acting without direct external intervention.
- Interactive - communicate with the environment and other agents.
- Adaptive - agents dynamically adapt to and learn about their environment. They are adaptive to uncertainty and change.
- Cooperative - able to coordinate with other agents to achieve common purpose.
- Social - they work together.
- Coordinative - able to perform some activity in a shared environment with other agents, via plans, work flows, or some other process mechanism.

An agent-based methodology is developed for building large-scale distributed systems with highly dynamic behaviors [33].A combination of intelligent agents and multi-agent approaches is applied to both local grid resource scheduling and global grid load balancing [34] and [35] applied use of economic agent in grid computing.

This chapter provides a conceptual framework for optimizing the cost of user or consumer by Bargain with the other agents based on their current state of the computing nodes.

4.2 Agent Description

The basic working definition of the agent is,it is an autonomous entity that can interact with its environment. In other words, it is anything that can be viewed as perceiving its environment through sensors, and acting upon on environment with effectors. Agent processing overview is shown in Fig. 4.1 [40].

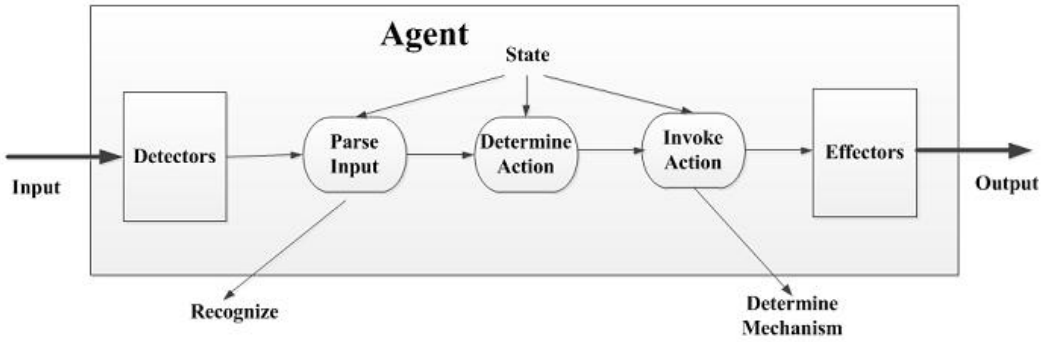

Figure 4.1: Agent processing overview

Agents may be autonomous and intelligent entities, which reside on nodes on a network, gets the problem from the users, discover resources and services, consult with each other, offer solutions and also learn from past experience and update their knowledge [39].

4.3 Introduction to pricing model

Distributed computing system is a collection of various computational resources with different owners and nodes. In economical point distributed market will be viewed as resource owners and resource consumers,a resource owner or consumer can define a resource usage policies like a task can run on a particular computing node only if it gives a certain revenue .

In addition to improving the performance of system, revenue of the owner or consumer will be improved by considering an economic pricing model [14, 41, 42]

that gives the cost benefits to the owner or consumer of any computing resource. The pricing policy proposed in this chapter is formulated using a agent based bargaining approach with an objective of determining the price per unit resource based on the current state of the other agents.

In any competitive market the decision related to weather to supply or consume resources mainly relies on the pricing strategies of the suppliers or consumers of the resources and this chapter formulates a dynamic pricing model with an objective of improving the revenue of the consumers and the price per unit resource is determined by the current state of computing nodes.

4.4 Need for pricing strategies

The distributed computing system is constructed by interconnecting computing nodes which are distributed across various locations, each computing node has its associated owner who is also called as resource supplier and the consumer or user uses those resources for execution of his tasks. The need for pricing strategies of resources owners or consumers arises due to

1. If consumers are accessing the resources, do the resource suppliers are charging any price for the resource usage ?
2. If resource owner are charging any price, do they charging same price for every user or do the every resource owner is charging the same price?
3. How the resource owners can earn more profit ?
4. How the users can execute their tasks with in the minimum cost ?
5. If resource owners can't finish tasks with in the deadline, do they reduce the price charged for tasks ?

By proposing the better pricing model there is a chance of getting profit for both the resource owners and consumers, while allocating the computational resources in distributed computing system. Already there exits numerous economic models for resource management are presented in the next section.

4.5 Economical Models

The most commonly used economic models for managing the computational resources in the distributed environment are

Commodity Market Model: [43]In the commodity market model, resource owners decide the price per unit resource for the amount of resources consumed. Here pricing policy depends on the supply and demand of the computational resources,in the flat pricing model once price of the computational resource is decided it is fixed for a certain period and it remains the same irrespective to the changes in demand or supply of the resources. Whereas in a supply-and-demand model price of the resources change very frequently based on the change in the supply or demand of the resources. Pricing schemes in a commodity market model can be based on

1. Flat fee
2. Resource usage duration
3. Demand and Supply of resources

Posted price model: This is almost same as the commodity market model, except that here resource owners advertises special offers to the consumer to attract new consumers in order to establish market using the cheaper slots.

Bargaining model: [43]In the previous models, the prices are fixed by the resource owners. In this bargaining model, resource brokers bargain with GSPs(Grid service providers) for lower access prices and higher usage durations. Both the brokers and GSPs(Grid resource providers) have their own objective functions and they negotiate with each other as long as their objectives are met. The brokers may start with a very low price and GSPs(Grid Service Provider) with a higher price.Both the resource owners and consumers negotiate until they reach a mutually agreeable price or one of them is not willing to negotiate any further. This negotiation is carried out by the user requirements (e.g., a deadline is too relaxed) and brokers can take risks and negotiate for cheaper prices as much as possible, and they can discard expensive machines.

Tender/contract-net model: [43] is one of the most widely used model for service negotiation in a distributed problem-solving environment.A user/resource broker asking for a task to be solved is called the manager and the resource that might be able to solve the task is called the potential contractor. From a manager perspective,the process is:

1. The consumer (broker) announces its requirements (using a deal template) and invites bids from GSPs(Grid service providers)
2. Interested GSPs(Grid service providers) evaluate the announcement and respond by submitting their bids
3. The broker evaluates and awards the contract to the most appropriate GSPs(Grid service providers).

From a contractor/GSP perspective, the process is:

1. Receive tender announcements/advertisements
2. Evaluate the service capability
3. Respond with a bid
4. Deliver service if a bid is accepted
5. Report results and bill the broker/user as per the usage and agreed bid

Bid-based proportional resource sharing model: [43, 44] In this model, the percentage of computational resource share allocated to the user application is proportional to that users bid value in comparison to the bids of other users. The users are allocated with credits or tokens, which they can use to access to computational resources. The value of each credit depends on the demand of the resource and the bid value of the other users placed on that resource at the time of usage. For example, consider two users wants to access a resource with similar requirements, the first user is willing to pay 2 credits and the second user is ready to pay 4 tokens. In this case, the first user gets one-third of

the resource share whereas the second user gets two-thirds of the resource share, which is proportional to the bid values that both users place on the resource for executing their applications.

Community/coalition/bartering/share holders model: [43, 44] A community of individuals shares each others resources to create a cooperative computing environment. Those who are contributing their computational resources to a common pool can get access to that pool. A sophisticated model can also be employed here for deciding how much resources share contributors can get. It can involve credits or tokens that one can earn by sharing a computational resource, which can then be used when they needed. A system like Mojonation.net employs this model for storage sharing. This model works when the participating entities in the Grid have to be both service providers and consumers.

4.6 Proposed Agent based Cost Model

An agent-based system designed to serve as matchmakers between the clients and servers in which client agent searches for the best servers for processing the jobs. The proposed scheme is modelled as an agent based approach, in which each computing node has as its associated agent which interacts with agents of the other computing nodes on behalf of the computing node.

This decentralized cost model consists of set of m independent computing nodes and each computing node has an its agent who is responsible for interaction with the other agents. In this price scheme agents are divided into two categories those are resource owners and resource consumers, where owner agents allocate computing resources to the tasks and they charge some price for task execution based on their processing capabilities and consumer willingness to pay and Consumers are the computing nodes who assigns tasks to the other computing nodes for execution, while mapping consumer agents bargain with owner agent regarding the price for task execution at that computing node.

In dynamic pricing scheme resource payments will change time to time based on the current state of the computing nodes and these current states of the

each computing node is tracked through the interaction between agents, based on these current state information agents negotiate with each other and decides the price per unit resource. Tasks arrive at different computing nodes may require different execution times based on their nature. On the other side, each computing node has a standard price for the resources to allow the execution of tasks. So the price between suppliers and consumers should be agreed by both sides otherwise they can bargain with each other for a new price based on the current state of the particular computing node. In this dynamic scheme resource payments will change time to time because in dynamic environment the workload at different computing nodes will vary.

The integration of the task scheduling system into this cost model has great influence in modelling a better cost model which gives the better revenue for the consumer. There are some parameters of the computational resources which effects the price per unit resource

1. Standard prices set by the resource suppliers.
2. Maximum price consumers willing to pay.
3. current state of the computational resource.

The proposed cost model has following steps

1. Every computing node has some standard price
2. When any agent is scheduling tasks to the other computing nodes,then this agent negotiates with other agents regarding price per unit resource.
3. Based on the current state of the system consumer agents bargained with the resource owner agents
4. Resource owner agents revise the price per unit resource based on the current system state and its processing capabilities
5. If both sides agree to the revised prices,then those revised prices will be used for task allocation, otherwise again they will go for negotiation.

This Bargain process is two-way,in which both the owner and the consumer agents negotiate with each for some new standard prices for resource allocation. In this bargaining process owner agent tries put the maximum price he can,at the same time consumer agent tries to pay as much as less price.

Here consumer agent argues that he is willing pay the price based on the waiting time at that particular computing resource. For example two computing nodes have waiting times 50 and 80, here customer bargains with the computing node 2, because of having more waiting time than computing node 1 he wants to pay less amount than the standard price. Based on the equation given below, consumer start bargain with the resource owners

$$BP(i) = SP(i) * \frac{Avg(W_{time}) - W_{time(i)}}{Avg(W_{time})} \tag{4.1}$$

At the same time owner agent argues that even though computing node 2 having more waiting time it can finish the task early as compared with computing node 1 because of having more processing power. So,owner agent will decide new price based on the ratio of processing power to the waiting time at that node and price decided here is less than standard price and greater than the bargained price of the consumer agent. The reason for setting the price between standard price and bargained price is, because of having more waiting time owner is compromising some price at the same time he tries minimize that compromised price by putting an argument that he can finish job earlier than the other computer instead of losing a customer. Based on the consumers bargain resource owner revises the price per unit resource which is economical to both the resource owner and consumers based on the current state of the system and its processing power.Revised prices will be calculated using the equation given below

$$RP(i) = SP(i) * (1 - \frac{\frac{pw(i)}{wt_i}}{\sum_{i=1}^{m} \frac{pw(i)}{wt_i}}) \tag{4.2}$$

Revenue of the consumer is given by the equation given below

$$Revenue = \sum_{i=1}^{m} [SP(i) - RP(i)] * P(i) \tag{4.3}$$

P is the vector specifying the number of tasks transfer to each computer from the computer i given by the OptimalFraction algorithm which is given in previous chapter.

4.7 Bargaining Algorithm and Description

In this non-cooperative bargaining approach before mapping tasks to other computing nodes, each computing node receive bids from other computing nodes which includes their standard prices and current waiting times at them. If computing node i is scheduling the tasks to other computers, before scheduling it will calculate new prices based on bids received and these new prices will be used for resource payments.

Algorithm 4 : Bargaining Algorithm

1: **procedure** BARGAIN
2: $[W_{time}, SP] = getCurrentState()$
3: **for** $i = 1 : 1 : m$ **do**
4: $BP_i = SP_i * \frac{Avg(W_{time}) - W_{time}(i)}{Avg(W_{time})}$
5: **end for**
6: $[(pw, W_{time}, BP] = getCurrentState()$
7: **for** $j = 1 : 1 : m$ **do**
8: $RP = SP(j) * 1 - \frac{\frac{pw(j)}{wt_j}}{\sum_{i=1}^{m} \frac{pw(i)}{wt_i}}$
9: **end for**
10: **end procedure**

Let SC is a vector specifying the Standard price to execute a task at different computers, these standard prices of the computing nodes are proportional to the processing powers of those computing nodes. Here these standard prices are generated by taking the relative processing powers of the computers and NC is a vector specifying the New price to execute a task at different computers which is calculated based on the Bargain algorithm given this chapter, here NC

is calculated based on the waiting times at different computing nodes.

Example:

SP=[20 30 40 20 50 60]

W_{time}=[6 10 5 3 20 10]

pw= [10 20 30 10 50 40]

Consumer Bargain

$BP(1) = SC(1) * \frac{54-6}{54} = 20 * 88\% = 18$

$BP(2) = SC(2) * \frac{54-10}{54} = 30 * 81\% = 24$

$BP(3) = SC(3) * \frac{54-5}{54} = 40 * 90\% = 36$

$BP(4) = SC(4) * \frac{54-3}{54} = 20 * 94\% = 19$

$BP(5) = SC(5) * \frac{54-20}{54} = 50 * 62\% = 31$

$BP(6) = SC(6) * \frac{54-10}{54} = 60 * 81\% = 49$

after simplifying bargained prices will be BP=[18 24 36 19 31 49]

Based on the consumer or user bargain resource owner will decide weather to revise the price or not. If the percentage of reduction in cost is more than his threshold value then resource owner will revise the prices. Let threshold value is 80%, means if the percentage in reduction is less than 80% then owner will revise price by arguing that eventhough he has more waiting time he can finish the execution earlier than others because of his high processing capability.

$RP = SP(5) * (1 - \frac{\frac{50}{20}}{23.8})$

$RP = SP(5) * (1 - \frac{2.5}{23.8})$

$RP = SP(5) * (1 - 0.12) = 50 * 88\% = 44$ Revised price for the computing resource 5 is 44, RP=[18 24 36 19 44 49]

let P=[0 5 3 0 1 1] means 5 tasks are assigning to computer 2,3 tasks are assigning to computer 3,1 task are assigning to computer 5 and 1 task are assigning to computer 6.

Revenue=[(30-24)*5 + (40-36)*3 + (50-44)*1 + (60-49)*1]=59

from the above example revenue of the user or consumer is 59 out of 290 means for

Number of Tasks	Standard Price	Bargained Price	Revenue
100	973	918	55
200	2664	2407	257
300	3237	2451	786
400	4766	4291	475
500	5464	4365	1099
600	6479	5897	582
700	7273	6384	889
800	9022	7047	1975
900	10088	8300	1788
1000	12179	9061	3188
Total Cost	62145	51155	11023

Table 4.1: Total Cost vs Tasks

the execution of 10 tasks at different computing nodes user will pay 321 instead of 380. In this example user is getting around 15% revenue.

From the table and graph Bargain algorithm is offering to pay less amount as compared with the standard prices.

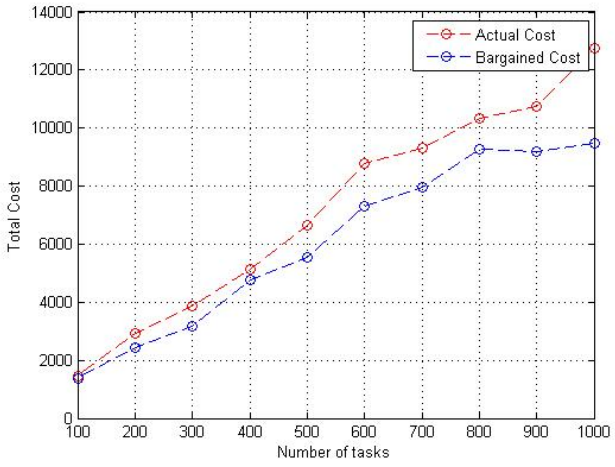

Figure 4.2: Total Cost vs Number of Tasks

From the above results Bargain algorithm is giving the average revenue around 18%,because before the task assignment it bargain with the other computing nodes regarding the price per unit resource, based on the current states of those computing nodes.

4.8 Conclusion

This chapter discussed about agent based framework for cost optimization, while allocating the resources with an objective of getting better revenue for user by using Bargain algorithm. Here resource allocation doesn't depends on this bargaining process. There are many studies discussed about price based resource allocation in which resources are allocated to the users based on the prices but here resource allocation doesn't depends on the price per unit resource. This proposed bargaining process use the resource allocation strategies to know the current state of the computing nodes based on this both the owner agent and user agent bargain with each other for getting better revenue.Based on this proposed pricing strategy, we can conclude that this Bargain algorithm is giving approximately 18% revenue to the consumers or users as compared with the standard prices.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

HDCS(Heterogeneous Distributed Computing System) is an environment for resource sharing and aggregation of distributed computational resources for solving complex computational problems. The computational resources in a distributed system are geographically distributed and each resource has its owner with different resource usage and pricing policies. This type of systems have large number of self-interested entities(resource owners and consumers) with different objectives that may vary from time to time. The management of computational resources in such a large and distributed environment is a complex problem.This thesis presented a dynamic non-cooperative resource allocation scheme for the management and regulation of on-demand resource supply.The proposed scheme provides a decentralized resource management capability and is adaptable to changes in the environment.

This work briefly discussed about the resource allocation problem as non-cooperative game among the computational nodes, with an objective of minimizing the makespan of the tasks arrive in the system by improving the fairness of the system and the effect of communication delay on makespan is also analysed under the hierarchical topology by varying the number of switches through which computing nodes are connected.

Another contribution is an agent based cost optimization with an objective of improving the revenue of the consumers or users of the distributed system. Most importantly computational cost of the tasks is minimized by paying the price based on the resource availability time(current state of the resource) instead of paying the standard prices. This approach provides economic incentives to resource owners by fair task scheduling and when the system is unfair user can pay reduced cost if deadline of the tasks are not critical.

5.2 Future Directions

In dynamic environment computational or communication resources will fail due to various reasons such as unavailability of required number of resources, overloaded resources, technical failure in the computational resources or communication link failure. In future work, our model can be implemented with different failures such as link failure or node failure.

Here resource allocation problem is studied by non-cooperative approach and it can be implemented by using the cooperative game theoretic approach.

Performance of this dynamic non-cooperative scheme is analysed under parameters makespan,fairness and effect of communication delay on makespan, in future the performance of this scheme is analysed under the performance parameters such as throughput,response time,flow time,success rate.

Some of the researchers have studied resource allocation with TIG(Task Interaction Graph). This thesis can be extended with task modelled as TIG for resource allocation on HDCS(Heterogeneous Distributed Computing System).

Resource allocation strategy can be further investigated considering energy consumed by under utilized computational resources in a Distributed Computing System(DCS). This can be possible with a heuristic to assign task to computing node so as to minimize the energy consumed by the task explicitly or implicitly [45].

Bibliography

[1] Xueyan Tang and S.T. Chanson. Optimizing static job scheduling in a network of heterogeneous computers. In *Parallel Processing, Proceedings. 2000 International Conference on*, pages 373–382, 2000.

[2] D. Grosu, A.T. Chronopoulos, and M.Y. Leung. Load balancing in distributed systems: An approach using cooperative games. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 52–61. IEEE, 2002.

[3] Daniel Grosu and Anthony T. Chronopoulos. Non-cooperative load balancing in distributed systems. *Journal of Parallel and Distributed Computing*, 65(9):1022 – 1034, 2005.

[4] M. Maheswaran, S. Ali, HJ Siegal, D. Hensgen, and R.F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth*, pages 30–44. IEEE, 1999.

[5] H.-C. Lin and C.S. Raghavendra. A dynamic load balancing policy with a central job dispatcher(lbc). In *Distributed Computing Systems, 1991., 11th International Conference on*, pages 264–271, May.

[6] Kenji Nishimura, Hitoshi Ueno, Miki Yamamoto, and Hiromasa Ikeda. A dynamic load balancing method based on network delay for large distributed systems. *Electronics and Communications in Japan (Part I: Communications)*, 84(6):11–21, 2001.

[7] R. Shah, B. Veeravalli, and M. Misra. On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments. *Parallel and Distributed Systems, IEEE Transactions on*, 18(12):1675–1686, 2007.

[8] Youran Lan and Ting Yu. A dynamic central scheduler load balancing mechanism. In *Computers and Communications, 1995., Conference Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on*, pages 734–740, Mar.

[9] M. Arora, S.K. Das, and R. Biswas. A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments. In *Parallel Processing Workshops, Proceedings. International Conference on*, pages 499–505, 2002.

[10] Issam Al-Azzoni and Douglas G Down. Decentralized load balancing for heterogeneous grids. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:*, pages 545–550. IEEE, 2009.

[11] P Neelakantan. An adaptive load sharing algorithm for heterogeneous distributed system.

[12] Martin J Osborne. *An Introduction to Game Theory*, volume 2. Oxford University Press, 2004.

[13] D. Grosu and A.T. Chronopoulos. A game-theory model and algorithm for load balancing in distributed system. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 164–153. IEEE, 2002.

[14] D. Grosu, A. T. Chronopoulos, and M. Y. Leung. Cooperative load balancing in distributed systems. *Concurr. Comput. : Pract. Exper.*, 20(16):1953–1976, November 2008.

[15] R. Subrata, A.Y. Zomaya, and B. Landfeldt. Game-theoretic approach for load balancing in computational grids. *Parallel and Distributed Systems, IEEE Transactions on*, 19(1):66–76, 2008.

[16] P. Ghosh, N. Roy, S.K. Das, and K. Basu. A game theory based pricing strategy for job allocation in mobile grids. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 82. IEEE, 2004.

[17] L. Kleinrock. Queueing systems – volume 1: Theory. new york:wiley. 1975.

[18] B.T. Doshi and H. Heffes. Overload performance of several processor queuing disciplines for the m/m/1 queue. *Communications, IEEE Transactions on*, 34(6):538–546, 1986.

[19] S. Ali, H.J. Siegel, M. Maheswaran, and D. Hensgen. Task execution time modeling for heterogeneous computing systems. pages 185 –199, 2000.

[20] F. De Rango, M. Tropea, A. Provato, A.F. Santamaria, and S. Marano. Minimum hop count and load balancing metrics based on ant behavior over hap mesh. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–6, 30 2008-Dec. 4.

[21] B. Janhavi, S. Surve, and S. Prabhu. Comparison of load balancing algorithms in a grid. In *Data Storage and Data Engineering (DSDE), 2010 International Conference on*, pages 20 –23, feb. 2010.

[22] K. Etminani and M. Naghibzadeh. A min-min max-min selective algorihtm for grid task scheduling. In *Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on*, pages 1–7, Sept.

[23] A. Shah and K. Kotecha. Efficient scheduling algorithms for real-time distributed systems. In *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on*, pages 44–48, Oct.

[24] J.F. Nash. *Non-cooperative Games*. Princeton University, 1950.

[25] T. Basar and G.J. Olsder. *Dynamic Noncooperative Game Theory*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1999.

[26] Eitan Altman, Hisao Kameda, and Yoshihisa Hosokawa. Nash equilibria in load balancing in distributed computer systems. *International Game Theory Review*, 4:91–100, 2002.

[27] S.S. Aote and MU Kharat. A game-theoretic model for dynamic load balancing in distributed systems. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*, pages 235–238. ACM, 2009.

[28] S. Dhakal, M.M. Hayat, J.E. Pezoa, C. Yang, and D.A. Bader. Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach. *Parallel and Distributed Systems, IEEE Transactions on*, 18(4):485–497, 2007.

[29] S. Penmatsa and A.T. Chronopoulos. Dynamic multi-user load balancing in distributed systems. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[30] Satish Penmatsa and Anthony T. Chronopoulos. Game-theoretic static load balancing for distributed systems. *Journal of Parallel and Distributed Computing*, 71(4):537 – 555, 2011.

[31] Satish Penmatsa and Anthony T. Chronopoulos. Job allocation schemes in computational grids based on cost optimization. pages 180a–180a, April.

[32] Xin Bai, Dan C Marinescu, Ladislau Bölöni, Howard Jay Siegel, Rose A Daley, I Wang, et al. A macroeconomic model for resource allocation in large-scale distributed systems. *Journal of parallel and distributed computing*, 68(2):182–199, 2008.

[33] Junwei Cao, Darren J Kerbyson, and Graham R Nudd. High performance service discovery in large-scale multi-agent and mobile-agent systems.

International Journal of Software Engineering and Knowledge Engineering, 11(05):621–641, 2001.

[34] Junwei Cao, Daniel P Spooner, Stephen A Jarvis, and Graham R Nudd. Grid load balancing using intelligent agents. *Future generation computer systems*, 21(1):135–149, 2005.

[35] Li Chunlin and Li Layuan. The use of economic agents under price driven mechanism in grid resource management. *Journal of Systems Architecture*, 50(9):521–535, 2004.

[36] P.K.K. Loh, Wen Jing Hsu, Cai Wentong, and N. Sriskanthan. How network topology affects dynamic loading balancing. *Parallel Distributed Technology: Systems Applications, IEEE*, 4(3):25–35, 1996.

[37] Xueyan Tang and S.T. Chanson. Optimizing static job scheduling in a network of heterogeneous computers. In *Parallel Processing, 2000. Proceedings. 2000 International Conference on*, pages 373–382, 2000.

[38] Yuan-Chieh Chow and Walter H. Kohler. Models for dynamic load balancing in a heterogeneous multiple processor system. *Computers, IEEE Transactions on*, C-28(5):354–361, May 1979.

[39] H. Homayounfar, F. Wang, and S. Areibi. Advanced p2p architecture using autonomous agents, 2002.

[40] Ashish virendra Chandak. Task scheduling strategies in grid. pages 47–54. National institute of Technology, Rourkela, 2012.

[41] Li Chunlin and Li Layuan. The use of economic agents under price driven mechanism in grid resource management. *J. Syst. Archit.*, 50(9):521–535, September 2004.

[42] John Von Neumann and Oskar Morgenstern. The theory of games and economic behavior. 1947.

[43] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[44] Rajkumar Buyya, Heinz Stockinger, Jonathan Giddy, and David Abramson. Economic models for management of resources in peer-to-peer and grid computing. In *ITCom 2001: International Symposium on the Convergence of IT and Communications*, pages 13–25. International Society for Optics and Photonics, 2001.

[45] Young Choon Lee and Albert Y Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.