AN AUCTION MECHANISM FOR GRID SCHEDULING AND RESOURCE ALLOCATION IN THE CONTEXT OF ATLAS

by

TENGKOK AARON THOR

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2009

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my committee chairs Dr. Gergely Záruba and Professor David Levine, who have been strong and supportive supervisors throughout my stay here as a doctoral student at the University of Texas at Arlington. They have spent countless hours providing me with valuable advices and guidance, both academic and in life. They have also been a tremendous resource in discussions about my research work, and have always encouraged my work and kept me focused on the important issues. Without their guidance and persistent help this dissertation would never have been possible.

I would like to thank my committee member, Dr. Manfred Huber, who has provided me with tremendous help on the formalization work, guidance, and advice on this dissertation. I would also like to thank Dr. Gergely Záruba, Professor David Levine, Dr. Kaushik De, Dr. Torre Wenaus, and Dr. Jae Yu for providing me with the great opportunity to be a part of the ATLAS experiment at Brookhaven National Lab. I would also like to thank Dr Kaushik De and Dr. Mark Sosebee for the years of support, advice, and encouragement they have provided for me. In addition, I would also like to thank my committee member Dr. Jeff Lei for his patience, encouragement, and understanding throughout the years.

I most want to thank my dad, Thor Kong Beng, my mom, Lee Chye Guek, and my sister, Shirlean Thor, for the years of faith, trust, encouragement and support. Thank you for believing in me.

July 10, 2009

ABSTRACT

AN AUCTION MECHANISM FOR GRID SCHEDULING
AND RESOURCE ALLOCATION IN
THE CONTEXT OF
ATLAS

Tengkok Aaron Thor, PhD

The University of Texas at Arlington, 2009

Supervising Professor: Gergely Záruba, David Levine

The technological advancements in the areas of computing and networking over the recent years have led to an emerging infrastructure known as Computational Grids, which provides users with the flexibility of pervasive access to enormous computational resources hosted at remote locations. Effective resource management and job scheduling poses a challenge when constraints such as resource utilization, response time, global and local policies need to be taken into account, while dealing with potentially independent sources of jobs, computational, and storage resources. It must be ensured that scheduling decisions made are still valid by the time a job is to be executed, with all the necessary resources remaining available.

In order to provide a more accurate scheduling and to obtain a better balance between micro and macro goals some status information about the resources needs to be obtained. However, this brings up another controversial issue which has plagued all dynamic scheduling communities: at what resolution monitoring should be performed. Since jobs are constantly

submitted throughout the grid and resources are used for processing such jobs, acquired monitoring information should be updated frequently. On the other hand, monitoring too frequently takes up valuable resources and bandwidth which could otherwise be used for job execution. Thus, another objective is to reach a balance between the risk of having outdated resource status information (which leads to incorrect scheduling decisions) and performing too much monitoring (wasting limited resources).

In a conventional grid environment, such as the ATLAS project [40], system administrators are often required to monitor the activities of a selected group of preferred sites and submit jobs to those sites if deemed capable of processing such tasks. The sites chosen, however, may not necessarily be the best sites for processing the jobs. This results in underutilized resources and stagnant efficiency of sites as there is no global incentive for improving efficiency as well as remaining competitive. One of the main reasons for sub-optimal resource allocation is that when taking factors such as system resource utilization, response time, global and local policies into account while dealing with potentially independent sources of jobs, computational and storage resources, the job of managing resources and job scheduling becomes too tedious for a centralized entity to perform. On top of that, with the implementation of varying local policies, a centralized scheduling entity may not have access or control over such policies, hence it could only perform scheduling based on a best effort basis. It is often up to the job-receiving host to perform the final leg of scheduling, based on its locally defined policies. As such, scheduling within a grid is often a multi-tiered process where the job-submitting host performs its job scheduling to the best of its knowledge of the current state of the grid environment, while the receiving host takes over the final phase of the scheduling process. Dividing the task of scheduling amongst several sites would add the advantage of easing the load and complexity of performing scheduling at a single location. However, in order to motivate individual local

domains in competing to become more efficient, in addition to being more aggressive in competing for accepting more jobs, some form of incentive mechanism could be applied.

In this work, we explore a decentralized combinatorial exchange scheme, as well as pull-based grid scheduling methodology which adopts the use of brokers with job advertisement and propagation within a grid environment. The main motivation for this scheme is to create an automated two tiered scheduling methodology to perform the tedious task of performing service discovery, and task scheduling at the global level, while performing resource monitoring, utilization and efficiency control at the local level. To achieve the best attainable optimization at any point in time, participating sites are to remain motivated to offer their best services based on the job submitting host's preferred optimization settings. Global scheduling of jobs will be done at the broker level via a bidding process. The submitting host will have the privilege to choose the best available offer to suit its requirements. A pricing scheme is implemented as a trading mechanism in exchange for the services provided. This pricing mechanism will hence serve as a motivation for participating sites to compete for jobs so as to increase its overall wealth. As such, competing sites will be required to constantly monitor and improve their own resources, its utilization and efficiency so as to remain competitive.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

LIST OF ABBREVIATIONS

CE	Combinatorial Exchange
CCE	Centralized Combinatorial Exchange
DCE	Decentralized Combinatorial Exchange
OCCE	One-shot Centralized Combinatorial Exchange
ODCE	One-shot Decentralized Combinatorial Exchange
RCCE	Repeated Centralized Combinatorial Exchange
RDCE	Repeated Decentralized Combinatorial Exchange
OCCCE	One-shot Centralized Closed Combinatorial Exchange
OCOCE	One-shot Centralized Open Combinatorial Exchange
ODCCE	One-shot Decentralized Closed Combinatorial Exchange
ODOCE	One-shot Decentralized Open Combinatorial Exchange
RCCCE	Repeated Centralized Closed Combinatorial Exchange
RCOCE	Repeated Centralized Open Combinatorial Exchange
RDCCE	Repeated Decentralized Closed Combinatorial Exchange
RDOCE	Repeated Decentralized Open Combinatorial Exchange

LIST OF SYMBOLS

$Aggc_i$	Consumer c_i's Aggressiveness Index
$Aggs_i$	Seller s_i's Aggressiveness Index
Ac	Consumer's auction bundle
${}^{all}Ac_i$	Set of all auction bundles requested by consumer c_i
${}^{all}Ac_{all}$	Set of all auction bundles requested by all consumers
As	Seller's auction bundle
${}^{all}As_i$	Set of all auction bundles offered by seller s_i
${}^{all}As_{all}$	Set of all auction bundles offered by sellers
b_i	Consumer i's budget
bi_i	Base incentive to execute job type i
bv	Seller's bundle valuation
C	Set containing all consumers
c_i	Consumer cost of acquiring resource bundle for job type i
c_p	Consumer cost of auction participation
c_π	Consumer penalty
$\widehat{cui}_i$	Consumer initial utility for job type i
$\widehat{cuf}_i$	Consumer final utility for job type i
$jea_i^{(t)}$	Job execution time actual (at time t)
$jee_i^{(t)}$	Job execution time estimated (at time t)
$jes_i^{(t)}$	Job execution time std dev (at time t)
jt	Job type
jv_i	Consumer job valuation of type i
jw	Job weight
$jw_{\max}$	Max job weight
$jw_{\min}$	Min job weight
ℓ	Lateness = $t_c - jee_i^t - tdl$

ρ	Resource market price
S	Set containing all sellers
s_{π}	Seller penalty
$\widehat{sui}$	Seller initial utility
$\widehat{suf}$	Seller final utility
t_c	Current time
${}^{A}\tau_{start,end}$	Expected time period usage for auction bundle *(A)*
tdl	Deadline time
$us_j^{(t)}$	Seller *j*'s resource utilization (at time *t*)
ut	Usage time of resource
Vc_i	Consumer c_i's Valuation
V_i	Consumer valuation of job type *i*
Vs_i	Seller s_i's Valuation
w_j	Seller j's wealth

CHAPTER 1

INTRODUCTION

The technological advancements in the areas of computing and networking over the recent years have led to an emerging infrastructure known as Computational Grids, which provides users with the flexibility of pervasive access to enormous computational resources hosted at remote locations. Effective resource management and job scheduling poses a challenge when constraints such as resource utilization, response time, global and local policies need to be taken into account, while dealing with potentially independent sources of jobs, computational, and storage resources. It must be ensured that scheduling decisions made are still valid by the time a job is to be executed, with all the necessary resources remaining available. However, resources in a grid environment are considered perishable goods, i.e., resources left unused cannot be “saved” for later use. Unused resources would not roll-over resulting in having an additional resource for use at a later time. In addition, incentives are often needed for individual local domains to strive for improved efficiency. This work will demonstrate the use of auctioning mechanisms and strategies for achieving the abovementioned objectives, along with the appropriate formulation methodologies for this application.

A computational cluster is a group of networked computers usually created by organizations for processing large data or computational intensive jobs. Several such (remote) clusters may be integrated together to form a computational grid. Thus, in a computational grid environment where data and computational intensive jobs are to be processed, there exists a vast collection of resources ready to process jobs of (virtual) organizations. Scheduling and resource management in a computational grid environment has been an area of extensive research due to its importance and complexity. Resources within a computational grid are likely to be heterogeneous, i.e., individual clusters or computers may be of different architectures, may be controlled by different operating systems and have diverse libraries. This often requires some form of resource matching such as [39] to effectively map jobs to resources. However, due to

fluctuating demands, resource availability may rapidly change. In addition, scheduling and management policies of clusters are unlikely to be uniform over all clusters in the grid. With ubiquitous resources distributed throughout clusters, it is important to be able to effectively manage these resources as well as assign jobs to take advantage of the available subset of resources. Without effective monitoring and management of resources, information on resource availability and the condition and duration of such availability is unknown. Furthermore, efficient scheduling is necessary in order to keep a particular resource from being overwhelmed when a similar resource may be "sitting" at another location idling.

Scheduling within a computational grid environment is often concerned with the welfare of the resources or components as a whole as well as the well being of individuals. By welfare of the resources or components as a whole, we often refer to such issues as fairness, e.g., equal opportunities to use resources, distribution of wealth throughout the grid and the overall performance of all the virtual organizations combined as a whole. Individual well being, on the other hand, is often concerned with the maximization of satisfaction derived from participating in work related activities. Such derived satisfaction may include minimization of response time and computation cost, while maximizing profit, throughput, and yield of earliest results. Thus, it is often necessary to be able to strike a balance between two contrasting goals: individual (per cluster) goals and system level utilization goals.

Individual (or local) goals, as the name implies, are more often concerned with maximizing the benefits individual entities can attain, with minimal regards for the welfare of the rest of the system. On the other hand, system level goals are usually more concerned with getting the most out of the currently available resources globally, even if it is at the cost of sacrificing a small population in order to benefit the system utility as a whole. For example, in order to maximize system resource utilization, a system level goal could be to keep all resources busy with a minimal number of unassigned resources idling. An individual goal, however, may be to have some idling resources available so as to have instant access to those resources when need arises. Similarly, it is a system level goal to ensure fair distribution of wealth (when completion of jobs is rewarded by some means) while an individual is more often interested in maximizing local

profits. As a result, it is a challenging task to satisfy individual needs, while still achieving system level goals at the same time.

In order to provide more accurate scheduling and to obtain a better balance between micro and macro goals some status information about the resources needs to be obtained. However, this brings up another controversial issue which has plagued all dynamic scheduling communities: at what resolution monitoring should be performed. Since jobs are constantly submitted throughout the grid and resources are used for processing such jobs, acquired monitoring information should be updated frequently. On the other hand, monitoring too frequently takes up valuable resources and bandwidth which could otherwise be used for job execution. Thus, another objective is to reach a balance between the risk of having outdated resource status information (which leads to incorrect scheduling decisions) and performing too much monitoring (wasting limited resources).

In a conventional grid environment, such as the ATLAS project [40], system administrators are often required to monitor the activities of a selected group of preferred sites and submit jobs to those sites if deemed capable of processing such tasks. The sites chosen, however, may not necessarily be the best sites for processing the jobs. This results in underutilized resources and stagnant efficiency of sites as there is no global incentive for improving efficiency as well as remaining competitive. One of the main reasons for sub-optimal resource allocation is that when taking factors such as system resource utilization, response time, and global and local policies into account while dealing with potentially independent sources of jobs and computational and storage resources, the job of managing resources and job scheduling becomes too tedious for a centralized entity to perform. On top of that, with the implementation of varying local policies, a centralized scheduling entity may not have access or control over such policies, hence it could only perform scheduling based on a best effort basis. It is often up to the job-receiving host to perform the final leg of scheduling, based on its locally defined policies. As such, scheduling within a grid is often a multi-tiered process where the job-submitting host performs its job scheduling to the best of its knowledge of the current state of the grid environment, while the receiving host takes over the final phase of the scheduling process.

Dividing the task of scheduling amongst several sites would add the advantage of easing the load and complexity of performing scheduling at a single location. However, in order to maintain high productive status, individual local domains have to constantly upgrade themselves and stay motivated in becoming more efficient, in addition to being more aggressive in competing for accepting more jobs, some form of incentive mechanism could be applied.

To address the problem of resource management and job scheduling in a large, geographically distributed network of virtual organizations, while observing the goals of cost minimization and improving utilization and efficiency, we propose a pull-based grid scheduling methodology which adopts the use of brokers with job advertisement and propagation within a grid environment. The main motivation for this scheme is to create an automated two tiered scheduling methodology to perform the tedious task of performing service discovery functionalities and task scheduling at the global level, while performing resource monitoring, utilization and efficiency control at the local level. To achieve the best attainable optimization at any point in time, participating sites are to remain motivated to offer their best services based on the job submitting host's preferred optimization settings. Global scheduling of jobs will be done at the broker level via a bidding process. The submitting host will have the privilege to choose the best available offer to suit its requirements. A pricing scheme is implemented as a trading mechanism in exchange for the services provided. This pricing mechanism will hence serve as a motivation for participating sites to compete for jobs so as to increase their overall wealth. As such, competing sites will be required to constantly monitor and improve their own resources, their utilization and efficiency to remain competitive.

1.1 Motivation

In a grid environment, resources are considered *perishable goods*, i.e., resources left unused cannot be "saved" for later use. Take for example the hotel industry; any rooms left empty for a night would be regarded as a wasted resource, since it would not roll-over, resulting in the hotel having an additional room for rent the following day (the total number of rooms is always fixed and remains the same each day). From the perspective of a resource provider, profit maximization through sales volume is often deemed as important as cost minimization, as any

additional units of unsold resources equate to additional incurrence of uncovered maintenance cost. This realization has led researchers to more closely investigate the economic behavior of grid systems.

Market based approaches have been gaining popularity in recent years [10][17], and are considered to work well in grid applications. Through these approaches, user preferences as well as provider compensations can be expressed efficiently in terms of costs, valuations and utilities. To date, many models have been proposed for the economics of grid systems through the adoption of fixed (e.g., [1][2][3][4][5][6][9][10][11]) and variable price (e.g., [12][0][14][15]) models, as well as different auctioning protocols (e.g., discussed in [2][3][4][5][6][7][8]). Strong assumptions are often required to simplify the price determination process in the abovementioned models. However, none of the above work has adequately formalized the price and utility determination for a market-based combinatorial exchange economic grid system.

When resource request matches are found, the matched resources are delegated from the resource owner to the resource requester. By delegating resources to jobs instead of the traditional migration of jobs to resources, we lower the administrative overhead of managing user/group accounts on each site where they can use resources.

1.2 Objectives And Contributions

The main objective of this work is the design, implementation, and evaluation of an economic grid mechanism that can meet scheduling and resource management requirements of a Grid environment. Prior to applying this methodology, we arrive at these specific questions.

What are the technical and economical characteristics of a market-oriented Grid mechanism?

In order to answer this question, the characteristics of Grid resources, requirements from its potential users, as well as the motivation behind resource providers to offer those resources have to be analyzed. The contribution is to find a link between the Grid and an economic environment where market-oriented Grid mechanism can be applied. This is performed through a

thorough formalization of the Grid environment including requirements, valuations, and compensations.

How to design a suitable mechanism for a market-oriented Grid?

The problem with most market mechanisms is that requirements are usually overly simplistic. Designing a mechanism for a market-oriented Grid would require one which is capable of expressing complex combinatorial requirements constraints seen in a Grid system while addressing the basic needs of a Grid environment. The contribution is designing and implementing one such hybrid mechanism which caters to all the requirements.

How to evaluate a market-oriented grid mechanism?

In order to evaluate a hybrid mechanism, a simulation model will be developed to provide an evaluation platform against working Grid systems. Real world data from the ATLAS experiment will be used as a benchmark for comparisons.

1.3 Chapters Outline

This work will be structured into three major parts, each with a focus on answering the three proposed questions. The first part will provide the fundamental understanding of both the Grid and Economic mechanism. The second part will discuss the design and implementation of a market oriented grid mechanism. Last but not least, the final part will evaluate the performance of the proposed mechanism against real world systems.

1.3.1. Part I

Chapter one will provide the introduction to the work, and chapter two provides the fundamentals of the Grid system. Chapter three will focus on the motivation for this work, and Chapter four provides background knowledge for the mechanism used to implement the work. Chapter five is divided into two sub parts. The first sub-part discusses previous work done in the

area of Grid scheduling and resource allocation, and the second sub-part discusses previous work on market-oriented Grid mechanisms.

1.3.2. Part II

Chapter six discusses the pricing mechanism used in our work, as well as defining the participant involvement in our system. Chapter seven discusses work done in the context of the ATLAS experiment [34], Chapter eight shows the formalization work, and also analyzes the applicability of different auction scenarios. Chapter nine focuses on the algorithmic implementation of your mechanism.

1.3.3. Part III

Chapter ten provides the design of our simulation while Chapter eleven analyzes the findings attained from our simulation results. Chapter twelve provides the summary and discusses future work for this market oriented Grid mechanism. Last but not least, Chapter thirteen will conclude the work.

CHAPTER 2

GRID COMPUTING FUNDAMENTALS

2.1 The Evolution of Grid

The early Grids started out as metacomputing, which was essentially an interconnection of several supercomputing sites. From the early to mid 1990s, the early metacomputing or grid started gaining popularity due to the need to tackle high performance applications such as FAFNER and IWAY [50]. FAFNER was designed as a parallel application, while IWAY was designed to work with high performance applications which required fast interconnect and powerful resources. After the success of the two projects further innovation in network technologies over the years has further helped in boosting the popularity of Grids. The second generation of grid advancement is credited to the vision of Grid in [51] which defined ways to cope with scale and heterogeneity in a Grid environment and the problems to be resolved in dealing with large scale computational power and information. Over the years, Grid is evolving towards the direction of a knowledge based economic grid, where resource and service providers offer on demand computing as services within the Grid environment.

2.2 Grid Architecture

The Grid follows a set of open standard protocols for message communication and controlling which was designed to form the basis for further interoperable development. The set of open standard protocols is shown in Figure 2.1. The protocols and interfaces are categorized depending on their function. The following details have been adopted from [20].

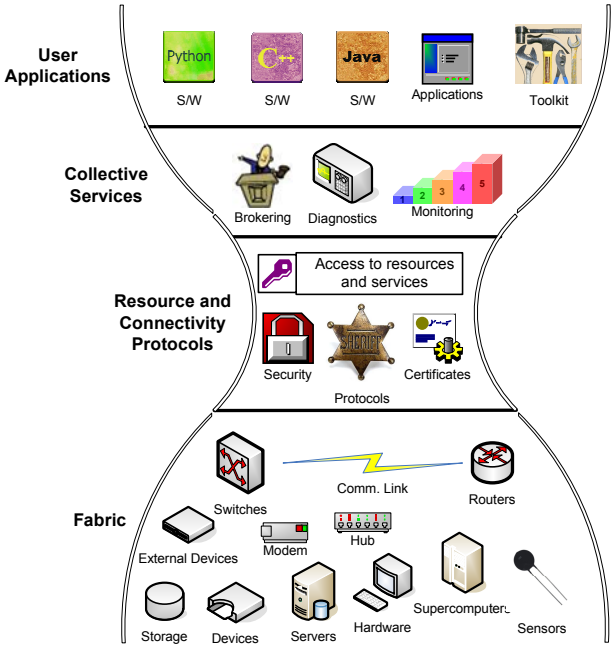

Figure 2.1 Layered Grid Architecture (Diagram format adopted from [20]).

Fabric layer: Jobs in this lowest layer are used to provide a common interface to all available resources. Access by higher layers is granted via standardized processes. All resources for which such a standardized interface is applicable, can be integrated in the grid concept. This contains computers, storage systems, networks or sensors.

Resource and connectivity protocols: The connectivity layer defines the basic communication and authentication protocols which are needed by the grid. While the communication protocols allow the exchange of files between different resources connected by the first layer, the authentication protocols allow to communicate confidentially and to ensure the identity of the two partners. To this belongs also the delegation of rights and methods for unique authentication (single sign-on). In the resources layer, the common access to individual resources is organized. This contains initiation, observation, control, clearance and negotiation of security parameters. Also, processor resources get assigned, reserved, observed and controlled. The OGSA is a standing architecture still in development that will lead the implementation of this layer in many grid projects. The Globus Toolkit 4 (GT4) presents a popular implementation of the OGSA

specification and offers software jobs and libraries to realize a grid according to OGSA specification.

Collective services: The purpose of this layer is the coordination of multiple resources. Access to these resources doesn't happen directly but merely via the underlying protocols and interfaces. The jobs of this layer contain, among others, the creation of a directory service, and they supply monitoring, diagnostic and file replication services. Furthermore grid-capable development systems are provided to be able to use popular programming models in a grid environment.

User applications: To this layer belong all those applications which are operating in the environment of a virtual organization. Jobs of the lower layers get called by applications and can use resources transparently.

2.3 Grid Resource Management

The term "resource management" is used to refer to the operations used to control how capabilities provided by grid resources and services are made available to other entities, e.g., users, applications, or services.

The main distinguishing factor between resource management in a grid environment and a traditional localized system is that resource management in grids may span across different administrative domains, usually operating under differing policies and conflicting objectives. In addition, resource management in grid environments often requires concurrent allocation and coordination of multiple resources across administrative domains. A key issue of resource management is deciding what resources to allocate to whom, and when. This includes the capability for resource discovery, resource scheduling and resource allocation.

2.4 Open Science Grid

OSG is a consortium of software, service and resource providers and researchers from universities, national laboratories and computing centers across the U.S., who together build and operate the OSG project. The project is funded by the NSF and DOE, and provides staff for managing various aspects of the OSG.

The goal at OSG is to bring petascale computing and storage resources into a uniform grid computing environment, and to integrate computing and storage resources from all over the U.S. and other countries.

The current bulk of resources consist of more than seventy participating institutions, including self-operated research VOs, campus grids, regional grids and OSG-operated VOs, capable of supporting about ten thousand CPU-days worth of processing in a day, with ten terabytes of data movement supported in a day.

2.5 Grid Applications

The Grid has a wide range of applications, such as solving challenge problems like protein folding, financial modeling, earthquake simulation, and climate/weather modeling. Grid also offers a means for providing resources as a utility for commercial and noncommercial users, with those users paying only for what they use, much like providing a service to consumers in a market.

Grid computing is also being applied to numerous scientific research projects and experiments around the world. One such experiment is the ATLAS experiment conducted at the Large Hadron Collider (LHC) at CERN, which is expected to produce over one hundred peta bytes of data over the next few years at the rate of around two hundred “events” per second, each event requiring approximately ten minutes of processing on a one GHz Intel processor.
Another well known project is Search for Extraterrestrial Intelligence (SETI) which was using more than three million computers to achieve 23.37 sustained teraflops (979 lifetime teraflops) as of September 2001[52].

CHAPTER 3

MOTIVATION FOR A MARKET ORIENTED GRID

To date, various mechanisms have been proposed to allocate resources in grid computing environments, but most of them have neglected one very important fundamental question in Grid computing: incentive. Most resource allocation algorithms have substituted valuations with job priorities, making the assumption that jobs with high valuation would naturally be assigned high priorities. This might be true for most cases, but failed to capture user preferences when using priorities. Job priorities refer to the weight assigned to jobs, which indirectly translates to the importance of that job perceived by a user. But it does not reflect the preferences of users for any set of resources. Some might propose the use of policies to differentiate user preferences, which might also work in this case. But how about incentives for resource providers to provide their resources for use? Policies may be able to distinguish between users' resource preferences, but cannot reflect the willingness of resource providers to offer their resources. In fact, resource providers might not be willing to provide their resources if there is no incentive for them to do so. Shifting the focus from placing weights and priorities on a per job basis to using valuations to represent users' valuation towards resources and resource providers' willingness to offer those resources, helps in achieving a much greater goal with just one variable instead of using combinations of weights, priorities, and policies for representing only user preferences and job priorities.

Moreover, incentive can also be translated to pricing models, penalties, discounts, and more, which provides a powerful and dynamic tool in manipulating the flow of resources within the Grid environment through the application of economic mechanisms.

CHAPTER 4

BACKGROUND

This section offers some background in the basics of the economic mechanism and various auctioning mechanisms available today.

4.1 Reasons for Markets

A market is a common place where trading transactions are executed. They exist mainly due to a fundamental rule in economics, the concepts of supply and demand. This could be described as a desire for something by some group, whereas another group (or groups) is offering that "exact something" in exchange for something else that the first group is in possession of or can acquire. Although this definition of a market is overly simplified and many would disagree with this overly generic explanation, it basically explains what a market is from a very abstract standpoint.

In order for a market to function, there has to be some kind of agreement for an exchange to take place. This is often known as market clearing in the classical school of economics. It is a simplifying assumption that exchanges always tend towards the price where the quantity supplied equals the quantity demanded, and this price is often called the market clearing price.

4.2 How Markets Work

There are many types of market clearing mechanisms in use today, many of which are designed to serve a specific purpose. As such, each has its own strengths and weaknesses for different applications. Figure 4.2 (partially adopted from [55]) depicts an overview of such market clearing mechanisms used. First off, trading price has to be determined prior to any trade to take place. Prices may either be static (fixed price) or dynamic in a market.

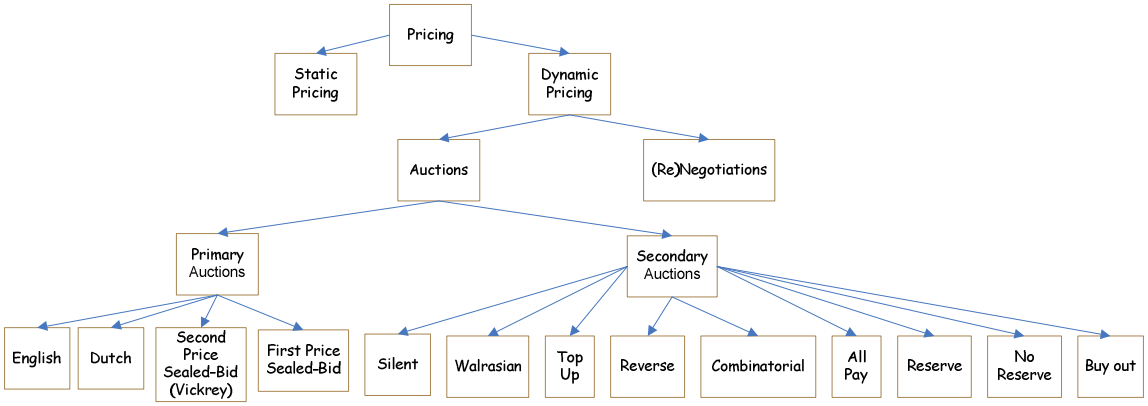

Figure 4.1 Market Types

Fixed prices, as the name implies, will never/seldom change during the course of a transaction. As such, this is more of a take-it-or-leave-it type of trade which limits further haggling on prices. The advantage of fixed pricing is in its simplicity and timely transactions. A trade is either successful or not, with no further complexities. For example, there are fixed price sales (such as what happens in a supermarket) where a consumer either purchases the product or walk away. This, however, may potentially limit the number of trades in a market due to its lack of flexibility. (This obviously does not mean that prices for this model can be determined in an ad hoc manner; indeed, the history of supply cost curves and demands for the products are used by economists to determine the price.)

Dynamic pricing, on the other hand, allows for some extended flexibility when compared to static pricing mechanisms. The price determination mechanism may be broken into two main groups: (1) Auctions, and (2) Negotiations / Re-negotiations. Auctions usually exhibit more flexible market clearing characteristics than fixed pricing, and are less complex and time consuming when compared to price negotiations. Since this work is focusing on auctioning mechanisms for grid resource exchanges, we will further discuss such pricing models in the oncoming subsection (Section 4.3). Price negotiations (one which allows for price haggling between sellers and consumers), are traditionally difficult to manage (due to the frequently non-policy driven interactive process between seller and buyer) and time consuming, especially those which allow re-negotiations. The frequency of negotiations plays an important role in the price determination process at the cost of further complexity and time delay.

4.3 Auctions

Auctions are often used as a mechanism to elicit information, in the form of bids, from potential buyers regarding their willingness to pay for an object (also known as the valuation of the object), and determine who-gets-what as an outcome. The objective of an auction is to achieve Pareto optimal outcomes, to attain allocative fairness and efficiency, or maximization of the seller's profit margin. From [53], Pareto efficiency or Pareto Optimality refers to situations where any change to make any person better off would result in having someone worse off. In other words, a Pareto efficient allocation refers to the best attainable allocation which cannot be improved any further. For more details on Pareto Optimality, readers are referred to [54]

Auctions are often used when a seller is unsure about the valuation of the item being sold from the perspective parties seeking to purchase the item. Should the precise valuation of every potential bidder be known to the seller, he could have simply sold the item directly to the potential customer with the highest valuation (or bid). On the other hand, bidders of an item often lack the knowledge of the valuation attached to the item by other bidders. In a best case scenario one may have an estimate of an item's worth based on an expert's appraisal or information collected from past experiences. But ultimately, bidders will still have to compete against one another in an attempt to win the item.

In a traditional auction, an auctioneer is regarded to be on the seller's side, taking bids from potential consumers who have valuations for the items they intend to purchase. In some auctions, however, an auctioneer agent may be on the consumer's side, taking offers from sellers who wish to sell their products. In our work, we make the assumption that auctioneers may be on the consumers' side (trying to minimize the final price), seller's side (trying to maximize the selling price), or neutral. When an auctioneer is on the consumer's side, its main objective is to maximize consumer's utility by maximizing valuations and minimizing purchase cost. If the auctioneer is on the seller's side, the objective would be to maximize profits while minimizing cost for the sellers. However, since both scenarios are considered to be one-sided (favoring either the consumers or sellers), a third neutral auctioneer scenario is introduced where the objective is to make an

attempt to strike a balance between the two prior cases with the main goal of maximizing global welfare.

4.3.1. Different Flavors of Auctions

There are many flavors of auctioning mechanisms, each with its own unique characteristics, strengths/weaknesses, and applications. Four primary types of auctions are widely used and analyzed, namely: ascending bid auction (often called open or English auction), the descending bid auction (also known as the Dutch auction), first-price sealed-bid auction, and second-price sealed-bid auction (widely known as the Vickrey auction).

In addition, there are nine secondary types of auctions used today, which are derivations of the primary auction types [58]:

I. **Silent auction:** is a closed-bid sealed-price auction where bids are historically placed by writing on a sheet of paper and putting this sheet in a sealed envelop (thus the name). The ultimate winner is the bidder who placed the highest price on her paper and the selling price is exactly this number.

II. **Walrasian auction (tâtonnement):** is a double auction where multiple sellers and consumers are matched in an auction based on bids taken from both sides in a market of multiple goods.

III. **Reverse auction:** as the name implies, reverses the role of sellers and consumers from a conventional auction where consumers compete and bid prices up in an attempt to win the auction. In a reverse auction, sellers compete by offering progressively lower prices until no supplier is willing to make a lower bid.

IV. **Combinatorial auction:** is an auction where bidders can place bids on combinations (or packages) of items, instead of being limited to bidding on single items like most conventional auctions.

V. **All-pay auction:** is a form of gambling on the outcome of an auction where participants must pay for the privilege of placing a bid in an auction-like process. Since the outcome of the auction-like process is uncertain, the amount spent on participating in the auction is similar to a wager, which is essentially a deceptive form of gambling.

VI. **Top up auction:** is a variation of an all-pay auction where bidders pay the difference between their current bid and the next lowest bid, regardless of whether they eventually win the auction. The winning bidder pays for the final item price while the other participants pay for the top-up prices.

VII. **Reserve auction:** is an auction where the seller sets a pre-determined minimum price for the item for sale. The item may not be sold if the final bid is not high enough to satisfy the seller's minimum price.

VIII. **No reserve auction:** is an auction in which the item for sale will be sold regardless of the final price. As such, the seller runs a risk of selling an item at a "ridiculously low price". On the other hand, psychologically, it has the advantage of attracting more bidders due to the inherent possibility of purchasing an item at a bargain.

IX. **Buy out auction:** is an auction with an initially set price (the 'buyout', or 'buy-it-now' price) that any bidder can accept at any time during the course of the auction, thereby immediately winning and ending the auction. If no bidder chooses to utilize the buyout option before the end of auction, the highest bidder wins the auction and pays their bid.

4.3.2. Single Auctions

Single-sided auctions can be categorized into:

I. Forward auctions, and

II. Reverse auctions.

(a) (b)

Figure 4.2 Single Sided Auctions.

Forward-auctions refer to a market where there is a single seller and multiple consumers competing for the product. Examples of forward-auctions are English and Dutch auctions. The design of forward-auctions favors the consumer with the purpose of revenue maximization. Reverse-auction, on the other hand, favors the purchase price minimization of a consumer.

4.3.3. Double Auctions

A double auction is an auction where consumers and sellers are treated as equals in an auction, with potential consumers and sellers submitting their bids concurrently to an auctioneer who will determine the best clearing price for the auction. Figure 4.3 shows the layout of a double auction scenario.

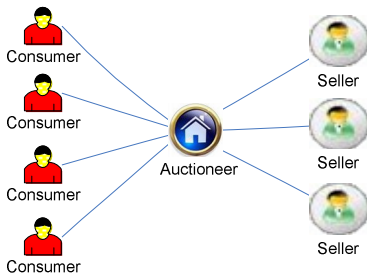

Figure 4.3 Double Auction.

4.3.4. Closed Auctions

In a closed auction bidders submit sealed bids. A participating bidder has no knowledge of the bids placed by other bidders.

4.3.5. Open Auctions

In an open auction, participating bidders have full knowledge of each other's previous bids and may repeatedly place higher bids using this knowledge.

4.3.6. Combinatorial Auctions

The definition for combinatorial auction has evolved over the years. As stated in [55], the classical variant of a CA is the multi-item auction where single items of multiple good types

are combined in bundle bids. It has later evolved to include multiple items of a good type in bid bundles, called multi-unit auction with combinatorial bids [57].

4.3.7. Combinatorial Exchange

Combinatorial exchange brings together both double auctions and combinatorial auctions into a single auctioning mechanism [56]. Buyers and sellers in a combinatorial exchange are able to trade single, multiple, homogeneous, or heterogeneous goods concurrently.

CHAPTER 5

PREVIOUS WORK

As the demand for computational power increases with the acceptance of computational grids, resource handling came under the spotlight as users began experiencing limitations in resource performance. What good can a tremendous amount of resources be when users are not utilizing them with caution? Careful scheduling and resource management eventually became one of the important goals of improving grid usage efficiency and unveiled important differences in resources between computational grids and in traditional computing systems. In a grid environment, resources are distributed and span across different administrative domains. Administrators of such domains are often not keen on letting someone from another domain take control of their resources. As such, one main obstacle to grid scheduling is to find ways to be able to achieve its scheduling and management objectives without taking full control of such distributed resources.

5.1 Traditional Grid Scheduling and Resource Management

In recent years, significant advancements have been witnessed in the area of grid scheduling. Although varying methodologies have been adopted, most shared a common focus on enhancing job performance and system utilization. As discussed in [47], grid scheduling can be classified into meta-scheduling and resource brokering as shown in Figure 5.1.

5.1.1 AppLeS (Application Level Scheduling)

AppLeS [47] is an application centric scheduling system designed with the objective of improving application performance in grids. In the AppLeS project [24], each grid application is tied with its AppLeS agent and scheduled according to its own performance model. Each AppLeS agent is made up of a coordinator and four subsystems: a resource selector, planner, performance estimator, and an actuator. The role of the central coordinator is to coordinate the

subsystems and perform the task of scheduling. The general strategy of AppLes is to take into account resource performance estimates to generate a plan for assigning file transfers to network links, and tasks (sequential jobs) to hosts. In recent years, AppLeS has begun development of AppLeS templates, where each template caters to a specific class of application.

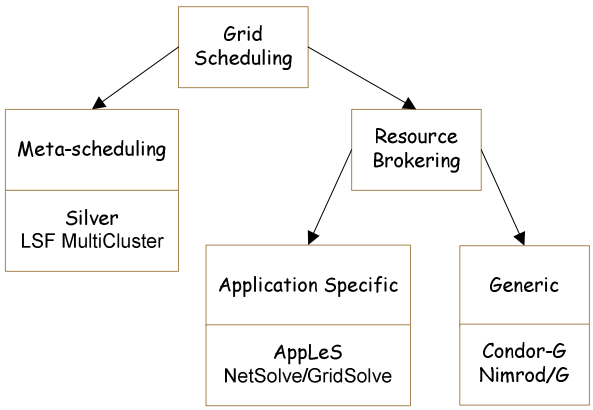

Figure 5.1 Taxonomy of Grid Scheduling (Adopted from [47]).

5.1.2 Condor

The Condor project [43] is a distributed computing research project conducted by the Computer Science department of The University of Wisconsin at Madison. It is an open source distributed computing software capable of handling large collections of distributed resources and job requests, providing a distributed high throughput computing (HTC) facility. Condor has the capability of handling both dedicated computing nodes as well as non-dedicated resources through cycle stealing. Condor's Globus Universe (i.e. Condor-G) is an extension allowing Condor tools to submit jobs to the grid. The Condor-G Matchmaking mechanism is used to schedule jobs to the grid. It allows users to specify requirements such as storage space, libraries, resource preferences etc. Figure 5.2 shows the architecture of Condor-G

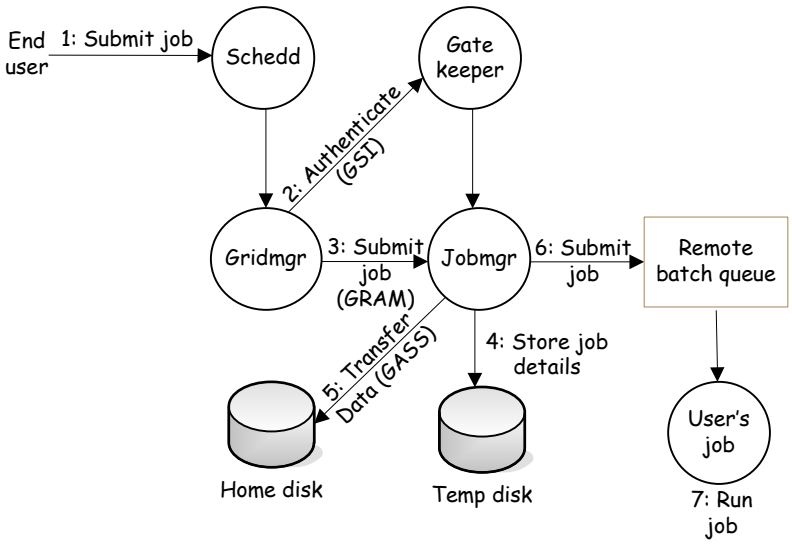

Figure 5.2 Condor-G (Adopted from [20]).

However, there have been several noted problems with using Condor-G as a grid scheduler. Although parallel jobs can be submitted in Condor-G, all jobs are treated as serial jobs since Condor-G does not support parallel jobs and it does not recognize the parallelism in jobs. In addition, although resource preferences can be defined in Condor-G, it will always favor closer datasets to explicitly defined resources because it does not support data aware scheduling. Scalability is also an issue in addition to the lack of integration with grid information systems. Another problem is the creation of orphan and wasted jobs in cases of failure at any key point, where jobs would be left running and consuming resources while never to be recovered or stopped.

5.1.3 Moab Grid Scheduler

Moab Grid Scheduler, also known as Silver, is a centralized grid scheduler encompassing features from several local schedulers such as PBSPro, Maui, and Loadleveler. It uses features such as advanced reservation, advanced co-allocation, and load balancing. It also provides support for jobs which span across multiple computing resources. The main

objective of Silver is to achieve optimal resource utilization, while providing flexibility in global and local policies and remaining simple to use and manage.

At a high level, Moab applies site policies and optimization techniques to handle jobs, services, and other workload across distributed grid resources. Moab Grid Scheduler also allows schedule reordering and resource allocation in an attempt to improve cluster performance and responsiveness. It uses advance reservations to reserve resources for use at guaranteed start times. In addition, Moab jobs can be modified based on different policies to help improve system utilization and minimize response time. Preemption is also supported for jobs with high priority.

5.1.4 Nimrod/G

Nimrod/G [49] was introduced by Buyya et. al as a first attempt to bring market economy based systems to the area of computational grid environments. Nimrod/G's brokering system has the capability of integrating various economic models into applicable areas of grid resource scheduling and management by providing resource users a way to specify resource requirements for various job types. Nimrod/G also has the flexibility of adopting different trading mechanisms such as auctions, bargaining mechanisms, and posted price models, based on information on current price and different policies. Auctioning mechanisms supported by Nimrod/G include bilateral bargaining and English auctions, albeit with limited support for advanced reservations and resource bundle trading.

5.2 Economic Grid Scheduling and Resource Management

Economic oriented scheduling and resource management mechanisms have been gaining popularity in recent years. A comprehensive study of market oriented grid applications can be found in [17].

Buyya et al. [44] investigate economy grids and requirements of economy based grid systems. The authors discuss ideas and challenges for implementing auction models for grid resource allocation, including the English, Dutch, and Double auctioning schemes. In the first

two schemes (English and Dutch auction), potential resource buyers bid for the right to use resources. In the Double auctioning scheme, sellers set the selling price of their resources and the buyers set their respective budget for resource purchase. A “middleman” (GMA) acts as a broker and matches the two participants if the prices meet. In the first two schemes, however, buyers compete for the right to use resources by placing higher bids, which limits the motivational aspects for the sellers to improve themselves in terms of efficiency and resource management in order to stay competitive. The double auctioning scheme attempts to split the competition between buyers and sellers; it is important to note that although this is a resource allocation scheme, the sellers are providing a service to those who have the purchasing power within the market. As a result, the pressure of staying competitive should fall on the sellers rather than those who are paying for a service. Moreover, creating competition amongst buyers would lead to deprivation of service for some potential buyers who cannot afford to pay high prices for a service, hence bringing down the number of potential trades between sellers and buyers. In addition, by shifting the competition to the sellers, there is an indirect advantage of motivating individual sellers to better improve themselves so as to stay competitive within the market, aiding in enhancing the quality of resource management throughout the grid environment.

Yeo et al. [11] proposed an extension to the system-centric cluster Resource Management Systems (RMS) to support utility-driven resource allocation and management by introducing four mechanisms: (I) Pricing, (II) Economy-based Admission Control, (III) Economy-based Resource Allocation, and (IV) Job Control. The Economy-based Admission Control unit primarily determines if a job is to be accepted based on the job details and QoS requirements. The Economy-based Resource Allocation unit performs the task of resource and job matching, along with dispatching jobs to the matched nodes. The Job Control unit assumes the role of monitoring for jobs and resources. In this scheme, jobs are submitted to the cluster RMS using user-level job submission specification where the admission control mechanism determines the

feasibility of accepting a job. If a job is deemed acceptable, the resource allocation mechanism would perform the necessary matching of the job to an execution node. However, if the job is not deemed acceptable by the admission control unit, the rejection decision would be fed back to the user. In other words, the responsibility of job submission lies upon the user to make smart decisions on where to submit the jobs to be processed such that the probability of getting an acceptance is high (much like what is employed in numerous conventional job assignments in the grid today). In a conventional scheme, the user attempts to make a smart decision as to where to submit the jobs and performs the job submission process. Although the targeted site may not necessarily be the best suited site for the job, the job is usually accepted by the receiving host unless its gateway is overwhelmed and starts dropping jobs. In this case the user would have to either resubmit the job at a later time or determine another potential execution site and attempt to send it there for execution. In this scheme, however, due to the potential rejection of jobs by the admission control mechanism, it would be necessary for the user to constantly monitor the status of each submitted job for rejection as well as monitoring all potential clusters in order to increase his chances of getting his jobs accepted.

Xiao et al. [45] present GridIS targeting incentive based grid scheduling focusing mainly on the aspects of aggressiveness in resource reservations. According to [45], the grid is essentially divided into resource consumers and providers, which may not always be the case in real systems as the resource consumer may also be a provider of other resources. The authors make two assumptions when designing the scheduling mechanism: i) execution time of all jobs is sufficiently long to make the overhead of remote job execution relatively negligible; ii) every provider is able to receive all job announcements. This works well in an environment with a small number of time consuming jobs to be executed in a fully connected network. However, depending on numerous factors such as availability of resources, status of a site, network topology, and cost of job execution at remote sites. It may sometimes be more efficient and economical to execute those jobs locally. In addition, although fairness in job distribution is an

attractive goal in grid computing, it often comes with the price of excessive network flooding and added delay from propagating job announcements throughout the network. As a result, it may be beneficial to have some mechanism for assessing overhead costs and evaluating propagation of job announcements to aid in determining the worthiness of sending a job for remote execution. In addition Xiao uses the ratio of agent payments versus resource price to calculate resource allocation for each agent. In reality, it may be more desirable to have a monitoring agent located at each participating site to handle the task of monitoring resource usage.

Opitz et al. [16] performed a thorough analysis of the various costs incurred by a resource provider in setting up and maintaining a grid resource center. They conducted analysis on several case studies and provided good estimates of costs in real grid systems. According to Opitz, costs can be classified into several categories (see Figure 5.3) based on what they relate to: hardware, premises, software, personnel, and data communication. One of the key observations is that certain costs become variable when components switch from idle to busy state. For example in today's technology, an idling system consumes substantially lower power when compared to the same system in a busy state.

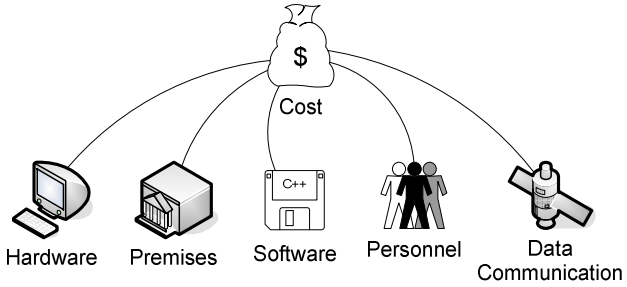

Figure 5.3 Operation Cost Breakdown.

Broberg et al. [17] provide a comprehensive evaluation of the current market-driven utility computing platforms. They categorize participants within such environment into three main groups: users, brokers, and service providers. One important key issue brought up by Broberg is that: "…the behavior exhibited in a shared system where market-driven techniques

are used simply to regulate access differs greatly from a profit-driven commercial system." [17]. In a profit-driven environment, service providers share a common goal of maximizing accumulated profit through the provisioning of their resources (usually with a secondary goal of incurring minimal cost). A profit-driven service provider might care less about global fairness and efficiency than the profit made from any one transaction. On the other hand, when adopting a market-driven mechanism with the objective of ensuring fair and unbiased access to resources, profit making might not be ranked as high as in a profit-driven environment. Thus the pricing techniques used in these fundamentally different systems might be vastly dissimilar.

Tan [46] proposed a Stable Continuous Double Auction (SCDA) scheme applicable to market-based grid resource allocation. The SCDA has the advantage of continuous matching, along with low communication and computation cost. It also provides low price volatility with low bidding complexity.

Tan made the assumptions that:

I. The grid environment is a distributed two-sided market with consumers and providers competing concurrently
II. The grid resource allocation mechanism must have the ability to offer resources and resource bundles with minimum delay
III. Discriminated price mechanism is used as the dynamic pricing mechanism
IV. Resource allocation and scheduling efficiency evaluation is based on economic efficiency (Pareto efficiency) and scheduling efficiency (user-centric performance).

The proposed SCDA scheme is a modified Continuous Double Auction (CDA) mechanism with an added Compulsory Bidding Adjustment (CBAL). The CBAL acts as a filter to perform price adjustments to correct unfavorable prices submitted by participants. All orders are channeled through the CBAL prior to reaching the standard CDA mechanism. Figure 5.4 shows the architecture of SCDA. The CBAL price adjustment mechanism uses a set of IF-THEN rules in a Mandani fuzzy controller in an attempt to translate price adjustment intuitions into fuzzy

rules which are then used to derive auction prices. Figure 5.5 shows the overview of the fuzzy logic used.

In the experiments conducted, five resource consumers and five providers were created for a grid environment, with provider cost prices generated from a uniform distribution of [1.0-9.0], whereas offer prices were similarly generated from the range [1.5-9.5].

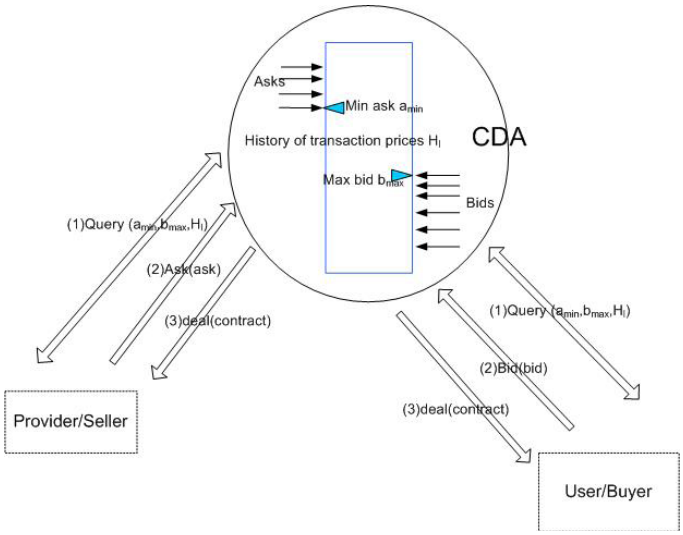

Figure 5.4 SCDA Architecture (Adopted from [46]).

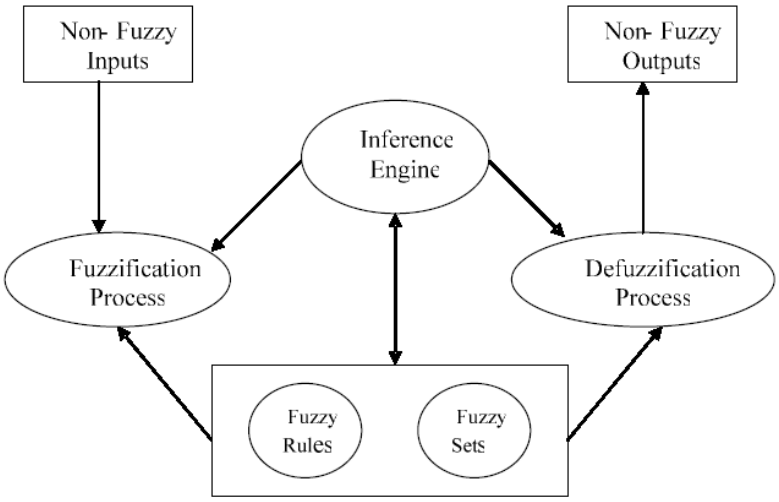

Figure 5.5 Fuzzy Logic Inference (Adopted from [46])

The use of Continuous Double Auction (CDA) has the advantage of being flexible and simple to implement, in addition to avoiding the computational complexities in the Winner Determination Problem (WDP). However, the author has conveniently disregarded one crucial rule in grid computing: resources in a grid environment are almost never used as standalone entities. For an auction mechanism to work in a grid environment, heterogeneous resource bundling capability has to be the de-facto standard if traded resources are to be of any use. Consider auctioning 2GB of memory without any CPU and storage support and the entire auctioning mechanism serves no purpose other than a classic buy-and-sell market for individual goods. On the other hand, it contributes to a new problem for consumers trading in such an auction environment – the exposure problem whereby winners are “exposed” to the risk of obtaining only a portion of a whole product, which is insufficient for any productive use due to the lacking of other required parts to make it a whole. As a result, the amount spent on acquiring the incomplete set of items goes to waste.

CHAPTER 6

PRICING MECHANISM

6.1 Supply and Demand

By definition, "…'Utility' is roughly synonymous with 'satisfaction,' 'well-being,' 'welfare', 'happiness,' 'pleasure,' etc. Generally, one can increase her utility by undertaking enjoyable activities or purchasing things we desire..." [18]. However, it is almost impossible to compute utility in closed form in the real world. Most researchers adopt the simplified route of defining utility by semi-arbitrarily assigning numerical values (with the use of $\prec$ notation [19] representing a preference function), or defining utility (u) as the difference between perceived value (v) and purchase price (p): $u(v, p) = v - p$.

Arbitrarily assigning numerical values is mainly used for representing an ideology as the resulting computations will not yield good accuracy. On the other hand, the use of differences between multiple perceived valuations introduces another term which is difficult to measure.

The work in [21] advocates the use of consumer surplus instead of the traditional utility representations. The advantages of using consumer surplus is that

I. it is measured in terms of actual currency (e.g., dollars),
II. since cost functions are also represented in terms of monetary values, it becomes much simpler to derive costs and to provide measurement of consumer satisfaction,
III. it enables the quantifiable measurement of aggregate consumer satisfaction in a given market

6.2 Market Pricing

Even if market demand is determined, no transactions can be realized without first setting a market price. In this section we will discuss the price determination process in a

market-driven grid environment as well as how a resource provider can determine the best price to sell and allocate their resources in a multiple market environment.

There are essentially three different types of costs [21] (see Figure 6.1):

I. sunk,

II. fixed, and

III. marginal cost.

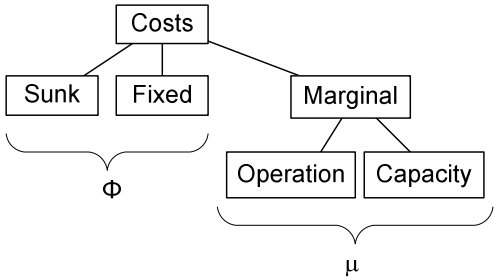

Figure 6.1 Production Cost Breakdown

Sunk and fixed costs are represented by Φ, while μ represents marginal costs. Sunk costs are costs which have already been spent. Any unspent costs can be divided into either fixed or marginal costs. Fixed costs in the economic grid environment refer to the costs incurred by the resource provider with the general operation of the resource center, e.g., acquisition of machines, storage disks, and setting up of networking are necessary prior to providing any services to any consumers. As a result, these can be classified as fixed costs. Marginal costs can be further categorized into marginal operation and capacity costs. Marginal operation is the cost incurred by proving service to one additional consumer, whereas marginal capacity cost refers to the cost associated with the cost of increasing capacity to attend to an additional consumer's resource requests.

6.3 Combining Goods

Combining goods in a sale can sometimes be the preferred method of marketing, due to the following reasons:

I. sellers may be able to maximize profit by selling goods in bulks,

II. the sales volume may be enhanced by selling more items in a single transaction.

Some marketing and economics literatures make a clear differentiation between the terms bundling and tying while others simply name them as bundling. For example, [21] defines bundling as selling of packages containing at least two units of the same product or service whereas tying is defined as selling packages containing at least two different products or services. In our work we will loosely use the term bundling to refer to bundling, tying, and combinations of both. In the following sections, we will explain how a seller decides whether to sell products individually or bundle them together as an entity. We also describe the different ways of bundling when attempting to maximize profit.

6.3.1. No Bundling

As the name implies, no bundling implies that all goods are marketed and sold as individual entities. Let V_A^{Cs1} denote the valuation of good *A* by a consumer *Cs1*, and V_B^{Cs2} represent the valuation of good *B* by consumer *Cs2*. Let P_A be the price of good *A*, P_B be the price of good *B*, and P_{AB} be the price for bundle *AB*. A potential consumer will only purchase an item if her valuation for that item is larger (or equal) than the selling price of that very item, i.e., if $V_A^{Cs1} \geq P_A$ then consumer *Cs1* will purchase good *A*; otherwise, *Cs1* is not buying *A*. However, if $V_A^{Cs1} \geq P_A$ and $V_B^{Cs1} \geq P_B$, then *Cs1* will purchase both goods *A* and *B*.

6.3.2. Pure Bundling

In pure bundling, goods are always sold in bundles and never as individual items. For example, assume that the price for good *A* is P_A, and the price of good *B* is P_B. In a pure bundling market, a consumer will either buy nothing, or both *A* and *B*, at the price of P_{AB}. From Figure 6.2, the area under the line P_{AB} represents consumers with combined valuation

$V_{AB} < P_{AB}$, thus nothing is being purchased. On the other hand, if $V_{AB} \geq P_{AB}$ (area above line P_{AB}), consumers will be willing to purchase A and B.

6.3.3. Mixed Bundling

In mixed bundling, goods may either be purchased as a bundle, or individually. Therefore consumers can choose among purchasing only good *A*, only good *B*, or both items *A* and *B*.

6.3.3. Multi-package Bundling

Multi-package bundling refers to offering different combinations of goods together as a bundle (adopted from [21]).

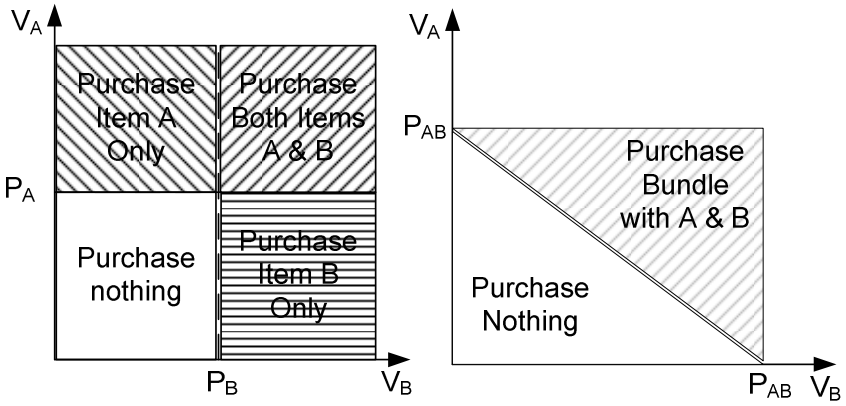

Figure 6.2 No Bundling vs. Pure Bundling.

6.4 Terminology

In this section, we explain the terminology used in our grid resource market model. The section is separated into two subsections:

I. Consumers, who are essentially users participating in the auctioning system in an attempt to attain computational resources for their job processing; and
II. Sellers, who are the providers of the grid resources through the auctioning system.

6.4.1. Consumers

Consumers (i.e., users) in a grid system by definition are trying to make the system process their computational jobs. Users attempt to harness the computational prowess of a grid to expedite their job processing capabilities. We define the set of consumers to be represented by *C*, where $|C| = c$ contains all consumers within the model.

6.4.1.1 Consumer Jobs

Every consumer has jobs which require the use of grid resources for their processing. The jobs in a grid environment are generally categorized into various job types. For example, in the ATLAS experiment [34], jobs are categorized into digitization, event generation, event analysis, reconstruction, simulation, etc. We define various types of jobs as $jt_1, jt_2, \cdots, jt_q$ where *q* is the number of distinct job types in the grid environment.

6.4.1.2 Consumer Job Weight/Priority

Since all jobs may not be created equal, some jobs may carry a higher priority than others. As such, every job within the system is assigned a weighting factor, e.g., job 1 is assigned weight jw_1, and all job weights are bounded by a lower and upper bound $jw_{\min}$ and $jw_{\max}$, respectively (they are predefined). For simplicity, throughout this work we can assume that $jw_{\min} = 1$ and $jw_{\max} = 100$ (unless defined otherwise). The use of a weighting factor helps in identifying priorities assigned to jobs.

6.4.1.3 Consumer Job Processing Time

In an ideal world, users know exactly how long each job will take to complete execution and hence could request resources for that exact amount of time. However, in the real world things often do not take place as planned and jobs can have stochastic behaviors. As a result, it is almost impossible to have a priori knowledge of the exact amount of processing time required for each job prior to execution completion. Users may, however, have some knowledge of the expected job execution time for each job type through experience. Although not necessarily

accurate, such knowledge is often useful when making estimates of when jobs will produce useful output. We define jee_i to represent the expected execution time of job type *i*, and jea_i to represent the actual execution time of job type *i*. In addition, the standard deviation (spread) of each job type's actual processing time is represented as jes_i. Both jea_i and jes_i are collected as historical information, and used when making estimations for the next expected job execution time.

Like most things in life, job behavior may change over time, due to changes in job characteristics, input and output dataset sizes etc. which in turn alters the average job execution time and standard deviation for each job type. In order to capture temporal job execution time characteristics, while minimizing the effects of any skewed data due to outliers, we will employ a temporal filter when determining jee_j^{t+1} at execution instance (discrete time) *t*, i.e., $jee_j^{t+1} = \sum_{i=1}^{t} \frac{jee_j^i}{t}$. This has the effect of evening out fluctuations in the latest job execution time, limiting outliers' skewing effects. jes_j^{t+1} is computed as $jes_j^{t+1} = \frac{1}{t-1} \sum_{i=1}^{t} \left(jee_j^i - \left(\frac{1}{t} \sum_{i=1}^{t} jee_j^i \right) \right)^2$.

6.4.1.4 Consumer Job Deadline

The job deadline is the latest time by which the job has to be completed without incurring additional penalty charges. It is usually estimated as the current time plus the expected estimation time plus one standard deviation and can be estimated using the formula: $tdl_i = t_c + jee^t(t_c, i) + jes^t(t_c, i)$ where tdl_i is the deadline for a job of type *I* and t_c is the current time.

6.4.1.5 Consumer Budget

In most grid job submission schemes, grid administrators can limit the number of jobs processed by each user on a per day/week/month basis. However, this limitation only serves as a cap to the number of jobs processed and offers no incentive for users to avoid wasting of grid resources. Budget serves a similar purpose of limiting resource usage, albeit with the additional benefit of being used in conjunction with valuation and utility as an incentive for consumers to curb excessive wastage of grid resources by always requesting top-of-the-line resources for processing non-critical jobs, including resources which could have been used for processing more critical jobs. Consumers of grid resources have a budget which they cannot exceed. This limits overly aggressive bidding behavior of consumers when participating in an auction for grid resources. Budget b_i refers to budget of consumer *i*.

6.4.1.6 Consumer Costs

When processing jobs, there is an inevitable cost associated with each job. The cost is the price paid to acquire the necessary resource bundles for job execution, and is calculated as $c_i = \rho * \left(jee_i + jes_i \right)$, where c_i is the cost of job type i. For example, the bundle price paid for the resources used to process a job. Since resources are acquired for an expected period of time in an auction, continued usage of the mentioned resources will incur additional charges, which is known as the penalty.

6.4.1.7 Consumer Job Valuation

In order for an auctioning model to work in a grid environment, there is a need for some type of incentive to entice consumers to participate in the auctions. Since users have the need to have their jobs executed, there is some 'value' attached to each job. This value function is expressed as the product of a base incentive value and the respective weight assigned to that job: $jv_i = jw_i * bi_i$ where jv_i is the job valuation of job type *i*, bi_i is the base incentive to execute job type *i*. Since a higher priority job would naturally be assigned a greater weight, its job value would also be higher than a lower weighing job. The consumer requires resource

bundles in order to process a job. Resource bundles are won from auctions by bidding on auction bundles (see Section 6.4.1.8). The expected value return from processing a job is the valuation of that auction bundle used to process the job, which is calculated as: $Vc_i\left(Ac_i\right)=\left[\left(\rho\right)*\left(jee_j+jes_j\right)\right]+jv_j$, where $Vc_i\left(Ac_i\right)$ is consumer c_i's valuation of auction bundle Ac_i. Note that this valuation can also be expressed in terms of cost, where $Vc_i\left(Ac_i\right)=c_i+jv_j$ (discussed in Section 6.4.1.6)

6.4.1.8 Consumer Resource Bundle Request

Consumers require grid resources in order to process a job. Different combinations of resources are bundled together and sold as an auction bundle. Consumers have to define the combination of resources required, expected time period needed of the resource bundle, and how much the consumer is willing to pay for that specified auction bundle. An auction bundle is defined as $Ac_i^t=\left({}^A x_{k,n},{}^A\tau_{start,end},Vc_i\left(Ac_i\right)\right)$, where Ac_i^t is the auction bundle consumer c_i submitted to an auctioneer at time *t*, ${}^A x_{k,n}$ is the resource combination specified in the auction bundle, ${}^A\tau_{start}$, ${}^A\tau_{end}$ are the start and end time period required of the auction bundle, and $Vc_i\left(Ac_i\right)$ is the maximum price consumer c_i is willing to pay for the auction bundle.

6.4.1.9 Consumer Auction Participation Fee

In order to participate in an auction, consumers are required to pay a small participation fee c_p for every bid placed. This is used to deter overly aggressive bidding behavior where consumers continuously place bids in an attempt to overwhelm the system so as to prevent other competitors from placing bids.

6.4.1.10 Consumer Penalty

The penalty is defined as the additional charges imposed when an agreement is breached by the consumer. For example, a penalty c_π is imposed on the consumer if she fails

to complete job processing before the deadline, or when the consumer's job fails to complete execution within the agreed upon period jee_i. The penalty is computed as follow: $c_\pi = max\left(\frac{\rho}{Us_j}, x\rho\right)$ where Us_j is the current resource utilization of seller s_j whose resources the consumer is using. ρ is the current market value of the resources used by the consumer. *x* is a scaling factor used to calculate the additional charges for overusing the resources.

6.4.1.11 Consumer Job Utility

The initial utility derived from winning a resource bundle for a job may sometimes be different from the final utility derived upon completion of a job since the initial utility is computed based on expected values whereas the final utility is the actual derived utility after job completion. The initial utility is $\widehat{cui}_j = Vc_i(A) - c_j + s_\pi - c_p$, where $Vc_i(A)$ is the derived valuation from acquiring auction bundle A (discussed in Section 6.4.1.7), c_j is the cost of job type *j* (see Section 6.4.1.6) which is the winning auction price for the resource bundle. s_π is the seller's penalty (discussed in Section 6.4.2.5) and c_p is the participation cost (discussed in Section 6.4.1.9). The final utility is computed as $\widehat{cuf}_i = \widehat{cui}_i - c_\pi$. ($c_\pi$ is discussed in Section 6.4.1.10)

6.4.1.12 Consumer Aggresiveness

The aggressiveness index of a consumer determines how aggressively she participates in the auction. It is computed as: $Aggc_i = \left[1 + \frac{jw}{jw_{\min} + jw_{\max}}\right]^{\frac{\ell * c_\pi}{Vc_i(A)}}$ where $\ell = t_c + jee_i^t - tdl_i$. The higher the aggressiveness index $Aggc_i$, the more aggressive the consumer would behave in her auction participation.

6.4.2. Sellers

Sellers participate in a grid auction to provide resources to consumers who require such resources to process their jobs. We define the set of sellers to be represented by S, where $|S| = s$.

6.4.2.1 Seller Wealth

In contrast to consumers which have a budget, every resource seller within a grid environment uses wealth for keeping track of her profits and spending. The ultimate goal of every seller is to maximize wealth while keeping spending to a minimum. Wealth w_j refers to the wealth of seller *j*.

6.4.2.2 Seller Valuation

Bundle valuation bv is the expected operating cost of the bundle of resources when used to process a job. Bundle valuation is calculated as $bv = \rho * (ut)$ where ρ is the market value of resource bundle, and ut is usage time of the resource bundle. Since resource sellers have no prior knowledge of how long a consumer will require the resource bundle, their bundle valuation is expressed in terms of price-per-unit-time. [16] provides a thorough study of the operating costs of grid resources.

6.4.2.3 Seller Utility

Similar to a consumer's utility computation, the initial utility derived from winning an auction by a resource seller may sometimes be different from the final utility derived upon completion of a consumer's job. The initial utility is $\widehat{sui} = P(A) - Vs_j(A) + c_\pi - c_p$, where $P(A)$ is the winning auction price for resource bundle *A*, $Vs_j(A)$ is valuation of auction bundle A by seller s_j. Since we assume here that sellers are willing to sell resource bundles at cost price, one can assume that $Vs_j(A) = bv$ (discussed in Section 6.4.2.2) here. c_π is the

penalty charges to a consumer (discussed in Section 6.4.1.10), c_p is discussed in the following Section 6.4.2.4. The final utility is: $\widehat{suf} = \widehat{sui} - s_\pi$, where s_π is the penalty charges paid by the seller (further discussed in Section 6.4.2.5)

6.4.2.4 Seller Participation Fee

Similar to consumers, sellers are also required to pay a small participation fee c_p for every bid placed, to deter overly aggressive bidding behavior where sellers continuously place ask bids in an attempt to overwhelm the system so as to prevent other competitors from competing.

6.4.2.5 Seller Penalty

Similar to consumer penalty, sellers pay a penalty when an agreement has been breached. For example, a penalty is imposed on the seller if she fails to provide the agreed upon resource bundle due to overselling of resources. The penalty is computed as follow:

$s_\pi = \max\left(x * P(A), x * \rho\right)$ where initial bundle price $P(A)$ is the agreed upon auction price for that bundle, and current market price ρ is the current market value of the resources offered by the seller. *x* is a scaling factor used to calculate the percentage of resource price to pay as penalty.

6.4.2.6 Seller Aggressiveness

The aggressiveness index of a seller determines how aggressive she participates in the auction. It is computed as: $Aggs_j = \begin{cases} \frac{1}{us_j} & us_j \geq 0.1 \\ 10 & us_j < 0.1 \end{cases}$

The higher the aggressiveness index, the more aggressive the seller behaves in the auction.

6.4.2.7 Seller Resource Bundle Offer

Sellers offer resource bundles in response to consumers' requests for resource bundles. These resource bundles are traded in auctions as auction bundles. A seller auction bundle is defined as $As_i^t = \left(us_j^t, {}^A x_{k,n}, {}^A\tau_{start,end}, Vs_i\left(As_i\right)\right)$, where As_i^t is the auction bundle seller s_i submitted to an auctioneer at time *t*, us_j^t is seller *j*'s resource utilization at time t, ${}^A x_{k,n}$ is the resource combination specified in the auction bundle, ${}^A\tau_{start}$, ${}^A\tau_{end}$ is the start and end time period required of the auction bundle, and $Vs_i\left(As_i\right)$ is the minimum price seller s_i is willing to accept for the auction bundle. Figure 6.3 shows the relationship between Consumer and Seller.

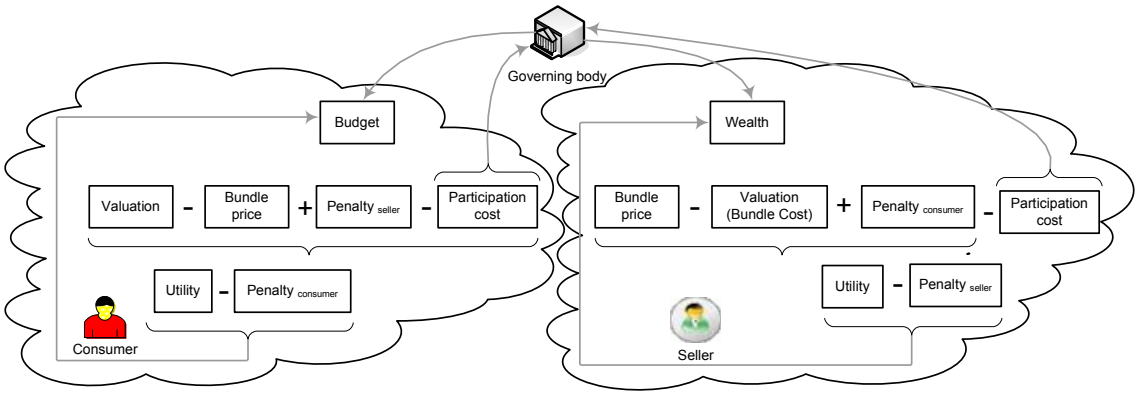

Figure 6.3 Consumer/Seller Relationship.

6.4.3. Historical data

Both consumers and sellers keep historical information on the following information:

- Past 3 resource prices for each job type. This is used to determine the directional trend of market pricing for each job type
- Number of counter bids placed by competitors, and their respective auction location, sorted order of decreasing number of counter bids order. This is used to identify the most aggressive competitors in an attempt to avoid direct conflicts in the future.

In addiction to that, each consumer keeps track of:

- All final utility returns $\widehat{cuf_i}$ from previous auctions, separated by job type.
- Past $Aggc_i$ values, sorted in ascending order.
- All final utility returns $\widehat{cuf_i}$ from transactions with all sellers, along with seller location, sorted in descending order.

On the other hand, each seller keeps track of:

- All final utility returns $\widehat{suf_i}$ from previous auctions, separated by job type.
- Past $Aggs_j$ values, sorted in ascending order.
- All final utility returns $\widehat{suf_i}$ from transactions with all consumers, along with consumer auction location, sorted in descending order.

CHAPTER 7

GRID MONITORING IN THE CONTEXT OF THE ATLAS EXPERIMENT

This section discusses work done on implementing a grid monitoring tool for the ATLAS experiment.

7.1 Grid Monitoring Overview

In order to support our study, we needed a simulation platform capable of supporting large scale grid environment while catering to the needs for an underlying networking infrastructure. As a result of these requirements, we created the ViGs simulator [33]. With ViGs, it allowed us to study the performance and behavioral aspects of the grid system as well as identifying any potential weakness within the system. The ViGs simulator possesses the ability to replace an entire functional grid environment, simulating all the resources and networking infrastructure which is used in a grid environment of today. To test the correctness of the ViGs simulator, we tested it against the PanDA (Production and Distributed Analysis System) [34] that is being developed by the US ATLAS (A Toroidal LHC ApparatuS) project with the goal of managing and scheduling very large workflow production and analysis of experimental results distributed across the US.

Production job summary, last 12 hours (Details: errors, nodes)

Cloud Information	Nodes	Jobs	Latest	Pilots (3hrs)	defined	assigned	waiting	activated	running	holding	transferring	finished	failed tot trf other
Overall Production	7908	81902	04-15 05:41	10807	0 / 0	2158 / 0	42 / 0	28922 / 0	14323 / 0	2731 / 0	14996 / 3464	16236 / 0	2534 / 19 14% 0% 13%
CA	909	9764	04-14 23:41	501	0	0	0	3692	1845	85	1008 / 0	3037	96 3% 0% 3%
DE	1219	14568	04-14 23:41	680	0	361	0	5391	2793	43	2020 / 0	3560	398 10% 0% 10%
ES	221	1980	04-14 23:40	32	0	0	0	1022	380	329	82 / 0	148	16 10% 0% 10%
FR	1802	15905	04-14 23:41	4886	0	462	0	3608	1971	1965	4121 / 1001	3514	263 7% 0% 7%
IT	201	2449	04-15 05:41	150	0	0	0	1275	87	6	458 / 63	548	75 12% 0% 12%
NL	266	7348	04-14 23:41	426	0	31	11	1425	565	11	3904 / 2368	205	1206 85% 0% 85%
UK	1716	11653	04-14 23:41	1248	0	1056	15	4796	2843	66	1179 / 31	1613	97 6% 2% 4%
US	1396	16435	04-14 23:41	2556	0	216	31	6990	3531	225	1972 / 0	3231	261 7% 1% 7%
TW	178	1800	04-14 23:40	328	0	32	0	723	308	1	252 / 1	380	103 21% 0% 21%

Figure 7.1 PanDA Production Summary.

7.1.1. PanDA Overview

A quick overview of the PanDA production system is shown in Figure 7.1. The PanDA production system is essentially a Many-Task Computing (MTC) 35] system consisting of numerous compute or data intensive tasks scheduled and processed on various computing resources spanning across multiple administrative boundaries around the world. It consists of the following components: ATLAS production interface, Regional usage interface, Monitoring system, Grid scheduler, PanDA server, and Participating sites, which includes research institutions around the world. Although we tried to capture the best representative 12-hour snapshot of the PanDA performance in Figure 7.2, the reader is reminded that a 12-hour representation of system performance is by no means a comprehensive representation of PanDA production performance. It serves to provide the reader with some basic information on the PanDA system, such as the total number of participating nodes at any point in time, how widely distributed the experiments are conducted (e.g. Canada (CA), France (FR), United Kingdom (UK), and United States (US) etc.), which stresses the importance of proper network modeling. Due to the various natures of jobs submitted, the typical job processing time ranges from ½ hour to 48 hours. Moreover, due to the nature of job locality 35], jobs often have the tendency to arrive in bursts. Hence, a period of 12 hours with more submitted short jobs might give the impression of superior system performance, whereas other periods where bursts of long jobs are submitted might give the false impression that the system performance has deteriorated considerably. In a typical PanDA operation, each node from a participating site submits what is called a pilot job request directly to the PanDA server. A pilot job can be seen as one possessing computational and data intensive characteristics of a MTC job which is scheduled and processed on computing resources crossing administrative boundaries. For the sake of simplicity, the inner workings of the PanDA server have been omitted. For more details on how the PanDA system works, readers are referred to 34]. Upon the completion of the input datasets' pre-placement process, jobs are delivered to the worker nodes prior to the initiation of

job execution (a typical LHC job takes from ½ hour to 48 hours to complete, depending on the job type). After jobs complete, the resulting dataset has to be "staged out" successfully to be tagged as 'finished'.

Figure 7.2 shows a snapshot of a PanDA production job summary over a 12 hour period. From the collected data, we can see that there were a total of 7908 active nodes available to process job, with 14323 running jobs and 16236 finished jobs within the 12 hour period. The failure rate was 14%, which is a little higher than its typical value of 10% for the PanDA production jobs. However, readers are reminded that this is only a snapshot of the PanDA summary over a period of 12 hours and it is by no means a complete representation of its performance in the long run. PanDA Production job performance may fluctuate due to unforeseen circumstances at times, as will be discussed in the next section. To get a better picture of the PanDA system performance, we have included plots of running, finished/failed jobs over a one-month period.

Figure 7.3 depicts typical running plots while Figure 7.4 shows a plot of finished and failed jobs over the same period. From the data collected in Figure 7.2, the averaged failure rate is calculated to be around 30% with a monthly job completion rate of 610,350 (for that particular month). The total number of jobs received over the period sums up to approximately 872,000 jobs, which averages to around 29,000 jobs per day. One of the main explanations for the observed differences between Figure 7.2 and Figure 7.3 can be attributed to the fact that since the PanDA system is still in the development and testing phase, factors such as upgrades (e.g. software and database upgrades), job outages (e.g. running out of jobs to fulfill pilot job requests), system instabilities (system, network, and power outages) etc. play an essential role in affecting the performance of the PanDA system. For instance, several snapshots of Figure 7.2 have been taken during the course of writing this paper, each with different characteristics. On a good day, the failure representation may go as low as 5~6%. On some rare occasions, however, the failure rate has gone as high as 47% with around 500 finished jobs.

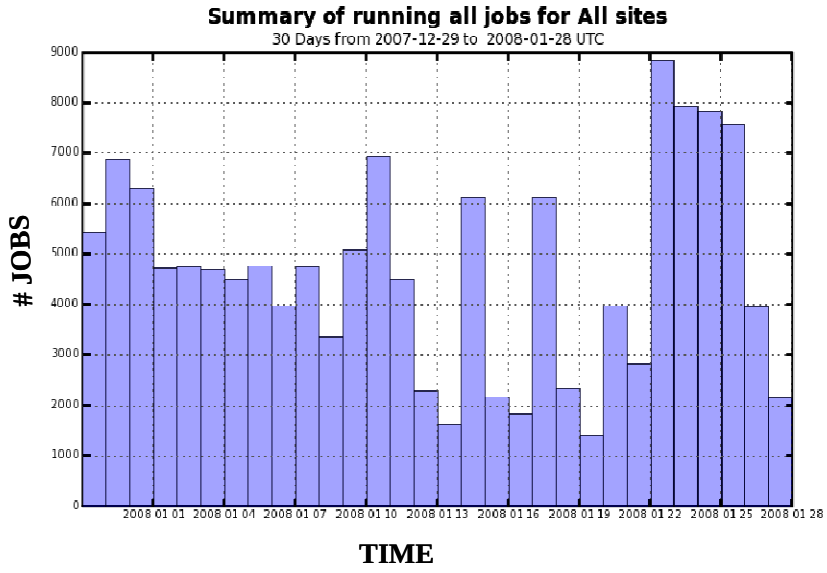

Figure 7.2 Summaries of PanDA Jobs.

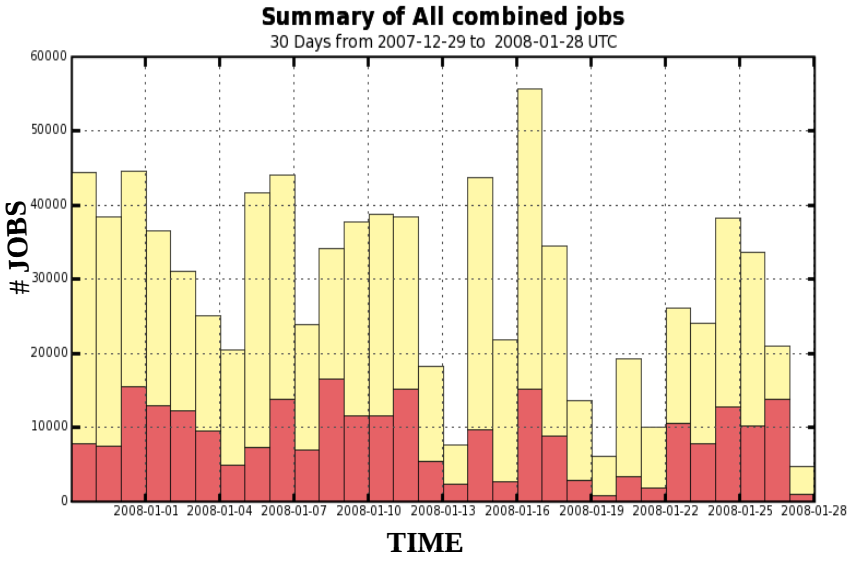

Figure 7.3 Summaries of PanDA Finished (light bars) and Failed (dark bars) Jobs.

7.1.2. Simulated Results

Figure 7.4 shows a typical PanDA server performance plot over a period of one month. From the collected data, we can observe that the PanDA server was able to consume an averaged total number of 1,134,650 jobs while producing completed jobs of approximately

872,000 within the one month time frame. The error rate for this period was computed to be 23.15%, which is relatively higher to what was observed from Figure 7.2 (at 14%), as well as our aggregated simulation results at 18.16%. One of the reasons for this slightly higher failure rate can be attributed to the intermittent networking problems which have been taking place at Brookhaven National Lab (BNL) where the current PanDA server is located. The slightly higher failure rates should gradually subside as the networking problems are resolved.

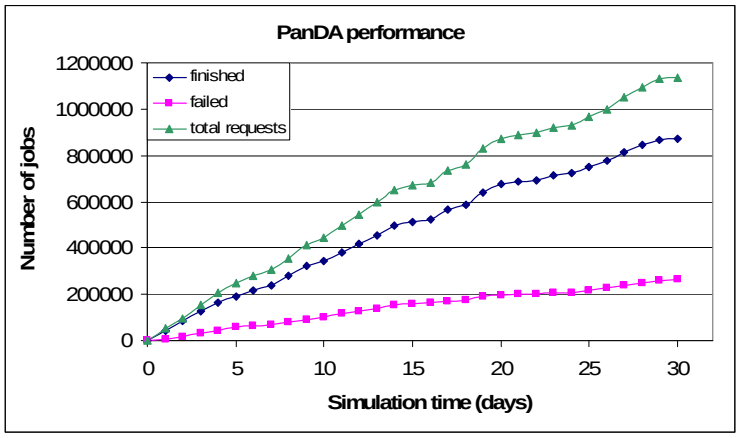

Figure 7.4 PanDA Server One-month Plot.

During the experimentation process, a total of thirty simulated 1-month experiments were conducted, each with a different seed feeding the ViGs simulator. Figure 7.5 shows the aggregated plot of our simulation results. When comparing the experimental findings, we noticed a consistent correlation between the two results: with the ViGs simulator running on a machine at full capacity, it was only able to consume about 80% of the pilot jobs when compared with the PanDA system. This may be attributed to the following reasons: On a typical day, the PanDA Production system has a number of active participating nodes ranging from six to eight thousand, many with two, four, or even eight core processors performing the job execution tasks. On the other hand, our ViGs simulator testing was performed on a single dedicated dual-core 2.4GHz machine with limited processing capabilities, limiting our processing threads to a maximum of 600. With the high resource utilization nature of this simulation, attempts to squeeze out more threads often resulted in instability to the system due

to overloading of the hardware. We believe that this phenomenon can be alleviated if we can create multiple instances of the ViGs simulator running in distributed mode. Figure 7.6 depicts the graphical comparisons of PanDA, PanDA at 80%, and ViGs simulator plots.

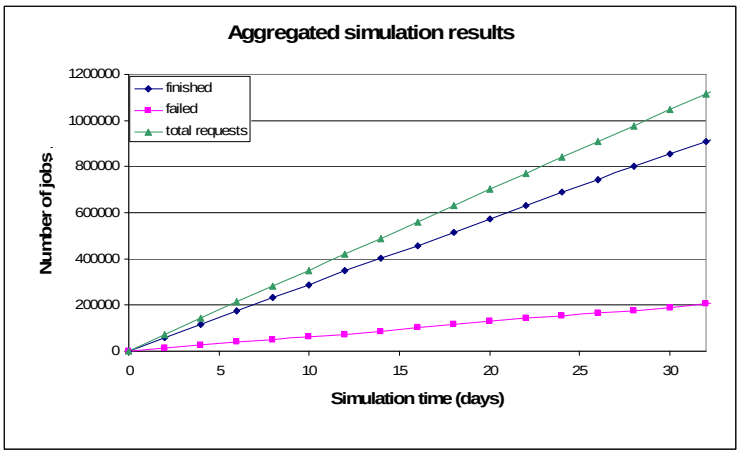

Figure 7.5 ViGs Simulation Results Plot.

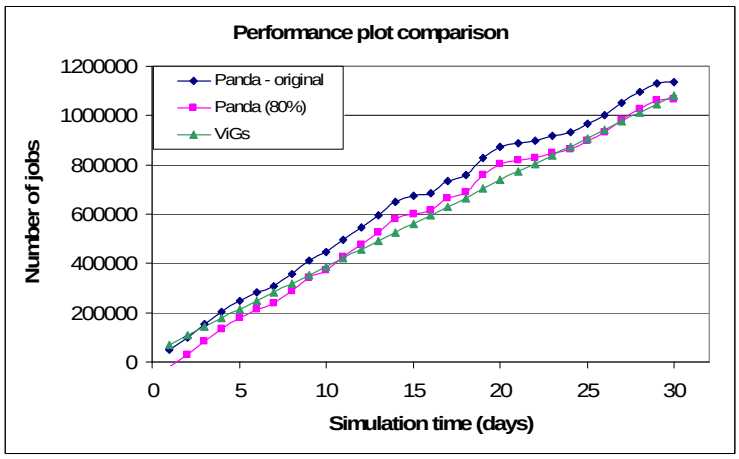

Figure 7.6 PanDA vs. ViGs Plot.

7.1.3. Observations

In this section, we make an attempt to find out how the results obtained from the ViGs simulator compare with the real-world production PanDA system. We perform our comparison by forming a confidence interval for the difference between production PanDA and ViGs simulation results. The Paired-t Confidence Interval approach [36] is used since it provides a good measure for comparing differences between the expected results of two systems by

pairing them together: For our case, real world performance results from the PanDA system and the simulation results obtained from the ViGs simulator.

We let X_j be the average of the observations in the j^{th} set of production PanDA data, and Y_j be the average of the observations in the j^{th} set of ViGs simulator resulting data, and $\mu x = E\left(X_j\right)$ and $\mu y = E\left(Y_j\right)$. Since we obtained a sample size of thirty results from our study, we let

$$Z_j = X_j - Y_j \Big\} \quad \text{for j} = 1..30 \qquad (1)$$

$$\text{and} \left.\begin{matrix} \mu_x = E(X_j) \\ \mu_y = E(Y_j) \end{matrix}\right\} \quad \text{for } j = 1..30 \qquad (2)$$

With $\mathrm{E(Z_j)} = \zeta$ as the value used to construct the confidence interval, we have:

$$E(Z_j) = \zeta = \mu_x - \mu_y \qquad (3)$$

By having $\left[l(\alpha), u(\alpha)\right]$ as the corresponding lower and upper confidence interval endpoints respectively, and using formula:

$$\bar{Z}(n) = \left.\frac{\sum_{\mathrm{j=1}}^{n} \mathrm{Z_j}}{\mathrm{n}}\right\} \quad \text{for j} = 1..\text{n, where n} = 30 \qquad (4)$$

$$\text{and } \widetilde{Var}\left[\bar{Z}(n)\right] = \frac{\sum_{\mathrm{j=1}}^{\mathrm{n}}\left[Z_j - \bar{Z}(n)\right]^2}{n(n-1)} \qquad (5) \quad ,$$

we can form an approximated 100 (1-α) percent confidence interval with:

$$\bar{Z}(n) \pm \mathrm{t}_{\mathrm{n-1,\ 1-}\alpha/2} \sqrt{\widetilde{Var}\left[\bar{Z}(n)\right]} \qquad (6)$$

Here, we attempt to compare the results obtained from the ViGs simulated model with the PanDA system by constructing a 95 percent confidence interval for ζ using the paired-t Test approach so as to determine if the model is a good representation of the system:

From (4): $$\bar{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}$$

$$= \frac{\sum_{j=1}^{30} Z_j}{30}$$

$$= 9.4615$$

From (5): $$\widetilde{Var}\left[\bar{Z}(n)\right] = \frac{\sum_{j=1}^{n}\left[Z_j - \bar{Z}(n)\right]^2}{n(n-1)}$$

$$= \frac{\sum_{j=1}^{30}\left[Z_j - \bar{Z}(n)\right]^2}{30(29)}$$

$$= 52.76675099$$

From (6): $\bar{Z}(n) \pm t_{n-1,\ 1-\alpha/2} \sqrt{\widetilde{Var}\left[\bar{Z}(n)\right]}$,

we get:

$$\bar{Z}(30) \pm t_{30-1,\ 1-\alpha/2} \sqrt{\widetilde{Var}\left[\bar{Z}(30)\right]},$$

resulting in:

$$-9.4615 \pm 2.045 * \sqrt{52.76675099}$$

where $$[l(\alpha), u(\alpha)] = [-25.8011785,\ 13.9088785]$$

Since $\bar{Z}(n)$ falls within interval $[\ l(\alpha)\ ,\ u(\alpha)\ ]$, with $0 \in [\ l(\alpha)\ ,\ u(\alpha)\ ]$, we conclude that the hypothesis is a good approximation, and the ViGs simulation result falls within approximately 95 percent confidence [36].

CHAPTER 8

GRID MARKET MECHANISM DESIGN

This section discusses work done on designing of grid market mechanisms.

8.1 Combinatorial Exchange (CE) Overview

Combinatorial Exchange (CE) differ from traditional auction models due to its inherent property that consumers can place bids on combinations (or bundles) of resources. This is useful when consumers require a combination of resources instead of only a single resource. Bundles of resources are essential in a computational grid environment where most jobs require multiple resources for their successful execution. For example, a typical job needs CPU for computational work, memory for virtual memory operation, and disk space for storage purposes. As such, a user wanting to process a job has to first acquire all the necessary resources prior to processing each job. If each resource is acquired individually, the user runs the risk of the "exposure problem", where the user is "exposed" to the risk of obtaining only a part of the necessary resources, which is insufficient for job processing due to the lacking of other required resources. As a result, the amount spent on acquiring the incomplete set of resources goes to waste. A CA asserts the fact that the consumer either successfully acquires all the necessary resources in a bundle, or gets nothing at all. This effectively avoids the possibility of an exposure problem. However, in a traditional CA, only one side of the participants (either sellers or consumers) may place bids in the auction. This renders market control to the participating bidders, and the non-bidding participants are forced to either accept what is being offered, or back out from participation.

In a Combinatorial Double Auction (CDA), multiple sellers and buyers are allowed to trade multiple units of resources in a single-sided auction. However, job processing in a grid environment often requires multiple users requesting combinations of heterogeneous resources

from multiple resource providers. Thus a CDA model does not meet our requirements for grid applications. On the other hand, a Combinatorial Exchange (CE) [26] auctioning environment allows both sellers and consumers to place bids on bundles of heterogeneous resources. In such an environment, no one party has full control of the market, and all participants are forced to compete against their respective peers in a truly competitive environment.

In a centralized auctioneer model (Figure 8.1), there is only one central auctioneer making the concluding resource assignment decisions based on the type of optimization used. In a decentralized auctioneer model (Figure 8.2), there are multiple auctioneers in the system, each hosting a separate localized auction.

In a closed bidding model, since consumers and sellers have no knowledge of the offers made by their respective peers, it would be more sensible to place bids based on their respective true valuation [27]. In an open bidding model, however, both consumers and sellers can take advantage of the added information to help in analyzing and determining subsequent bidding strategies in order to remain competitive in the market.

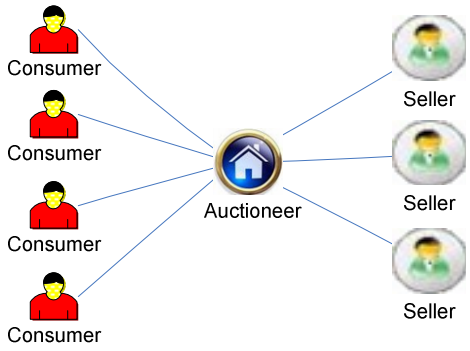

Figure 8.1 Centralized Auctioneer.

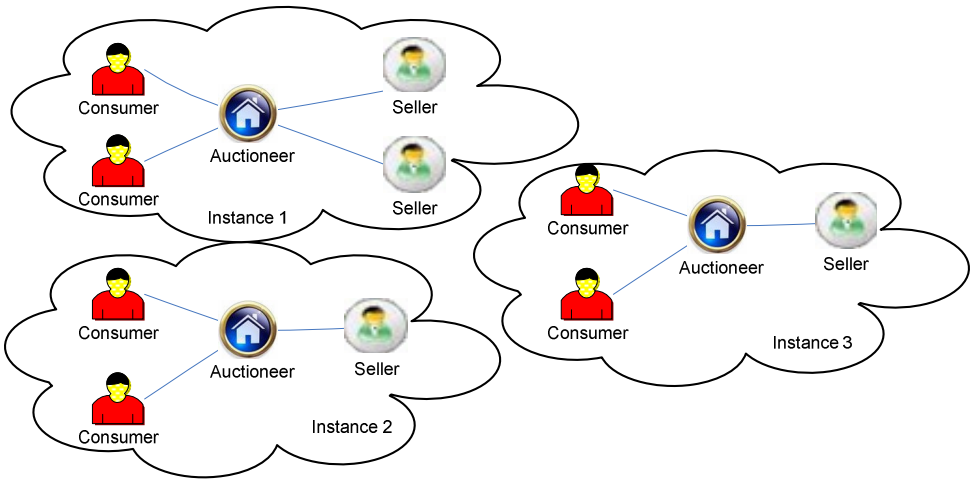

Figure 8.2 Decentralized Auctioneer.

A one-shot [28] auction model has only one trading period. So auctioneers will only make the consumer-seller matching once at the end of the auction. It is much like a 'memory-less' auctioning model where all auction participants have no historical information of previous market behavior. On the other hand, a repeated [28] auction model consists of multiple trading periods, where auctioneers will make the consumer-seller matching once at the end of every auction. We make the assumption that participating bidders have historical information from prior auctions, thus allowing them to make better bidding decisions based on perceived market trends. Figure 8.3 shows the different auctioning models which will be discussed and formalized in this chapter.

From Section 6.4.1.5, budget b is the set representing each consumer's budget. For every purchases made by each consumer, the amount spent has to be within budget for every consumer. Wealth $\hat{w}$ is the set representing each seller's wealth. Every seller's objective is to maximize their collection of wealth.

Let the set of all globally available unique resources be represented by M, where |M|=m. From the definition of bundle in Section 6.3, a bundle of resources can be represented by a set containing multisets of resources.

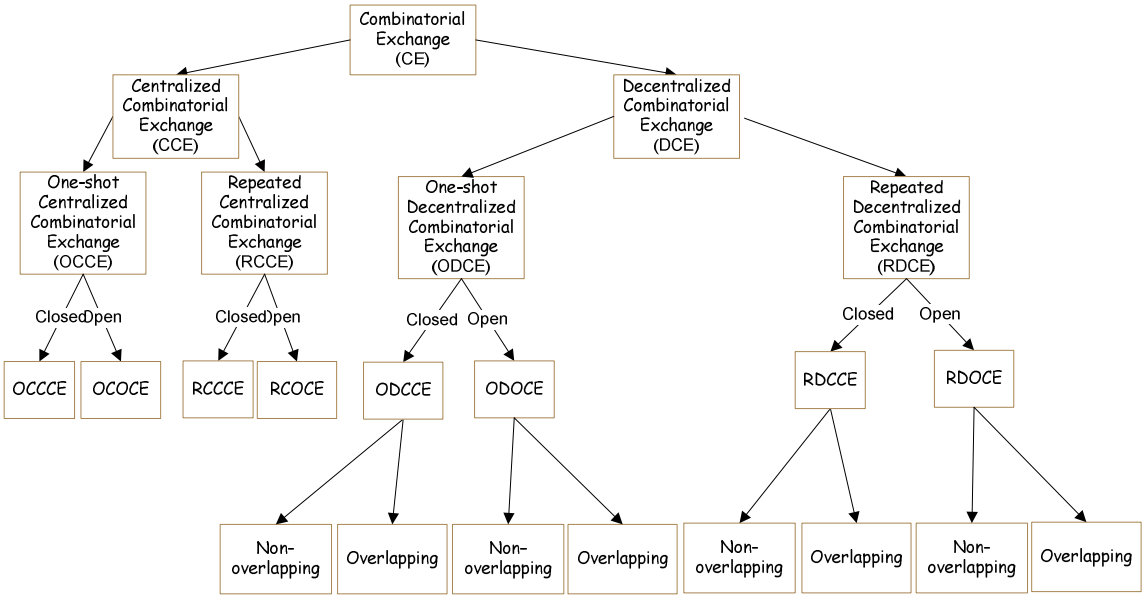

Figure 8.3 Auction Model Overview.

From [22], "The number of multisets of cardinality k, with elements taken from a finite set of cardinality n, is called the multiset coefficient $\left(\!\binom{n}{k}\!\right)$, where $\left(\!\binom{n}{k}\!\right) = \frac{n(n+1)(n+2)\ldots(n+k-1)}{k!} = \binom{n+k-1}{k} = \binom{n+k-1}{n-1}$".

The term $\left(\!\binom{n}{k}\!\right)$ can be represented in the form: $\frac{\prod_{i=1}^{k} n+i-1}{\prod_{i=1}^{k} i} = \prod_{i=1}^{k} \frac{n+i-1}{i}$

From [23], $x \in^{n} y$ is defined as "x is an element of y with multiplicity n"

For example, if $m = \{a,b\}$, there are $\prod_{i=1}^{3} \frac{2+i-1}{i} = 4$ multisets of cardinality 3, namely: $\{a,a,a\}$, $\{a,a,b\}$, $\{a,b,b\}$, *and* $\{b,b,b\}$.

Let *X* represent a set consisting of all possible multisets of m resources, where $X = \{ x_{1,1},..,x_{k,1},x_{1,2},..,x_{k,2},...,x_{k,n} \},\ with\ n = \prod_{i=1}^{k} \frac{m+i-1}{i}, and\ k,n \in \mathbb{N}$.

k = maximum number of resources in a bundle,

n = total number of possible unique k-element bundles.

The size of bundle *X* is represented by $|X| = \sum_{k=1}^{n} \left(\prod_{i=1}^{k} \frac{m+i-1}{i} \right)$. For example, given $m = \{a,b\}$, the set of possible multisets of m resources is shown in Table 8.1, where $X = \{x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, \cdots, x_{5,6}, \cdots x_{k,n}\}$. From the table, resource bundle 'aaabb' may either be represented in terms of $x_{k,n}$ (as $x_{5,3}$) or in terms of $x_{k,n}^{count} x_{k,n}^{count} \cdots x_{k,n}^{count}$ (as $x_{1,1}^{3} x_{1,2}^{2}$ where $x_{1,1}^{3}$="aaa" and $x_{1,2}^{2}$ = "bb").

Table 8.1 Multiset Representation for m Resources.

	k=1	**k=2**	**k=3**	**k=4**	**k=5**	**…**
n=1	a	aa	aaa	aaaa	aaaaa	
n=2	b	ab	aab	aaab	aaaab	
n=3		bb	abb	aabb	aaabb	
n=4			bbb	abbb	aabbb	
n=5				bbbb	abbbb	
n=6					bbbbb	

A consumer c_i can request for any auction bundle in an auction by specifying $Ac_i^t = \{ {}^{A}x_{k,n}, {}^{A}\tau_s, {}^{A}\tau_e, {}^{A}V \}$, where t is the auction bundle submission time, ${}^{A}x_{k,n}$ is the desired bundle resources for auction bundle A, and $Vc_i(Ac_i)$ defines the valuation consumer c_i placed on auction bundle Ac_i. Similarly, a seller s_j may offer any auction bundle of resources by specifying $As_j^t = \langle {}^{A}us_j^t, {}^{A}x_{k,n}, {}^{A}\tau_s, {}^{A}\tau_e, {}^{A}V \rangle$. Where us_j^t (discussed in Section 6.4.2.7) is seller j's resource utilization at time t, and t is the auction bundle submission

time, ${}^{A}\tau_s$ and ${}^{A}\tau_e$ defines the start and end time specification, respectively, for resource bundle $x_{k,n}$ in auction bundle A. It may be either the time period requested by a consumer for a resource bundle, where start is the expected job execution start time and end is the job deadline; or the time period availability for a resource bundle offered by a seller. By default, start is always zero (from consumers) for jobs which are to be executed immediately, or when resource bundles are available immediately (from sellers). A non-zero start parameter is used only for advance reservation of resource bundles. We define operator functions where:

$${}^{A}x\left(Ac_i^{\,t}\right) = {}^{A}x_{k,n} \quad \textit{resource bundle requested}$$
$${}^{A}\tau_s\left(Ac_i^{\,t}\right) = {}^{A}\tau_s \quad \textit{start time}$$
$${}^{A}\tau_e\left(Ac_i^{\,t}\right) = {}^{A}\tau_e \quad \textit{end time}$$
$${}^{A}V\left(Ac_i^{\,t}\right) = {}^{A}V \quad \textit{auction bundle valuation}$$
$$c\left({}^{A}x_{k,n}\right) = c_i \quad \textit{consumer requesting auction bundle containing resource } {}^{A}x_{k,n}$$
$$s\left({}^{A}x_{k,n}\right) = s_i \quad \textit{seller offering auction bundle containing resource } {}^{A}x_{k,n}$$

Both consumers and sellers may request/offer multiple bundles of auction bundles during an auction. Let $\left(Ac_i\right)_h$ denote the h^{th} auction bundle requested by consumer c_i. Thus all of consumer c_i's auction bundle request may be represented as: $\sum_{\ell=1}^{h}\left(Ac_i\right)_\ell$

Let ${}^{all}Ac_i\left(where\ {}^{all}Ac_i = \biguplus_{\ell=1}^{h}\left\{\left(Ac_i\right)_\ell\right\}\right)$ represent the set of all auction bundles requested by consumer c_i, and ${}^{all}As_j\left(where\ {}^{all}As_j = \biguplus_{\ell=1}^{h}\left\{\left(As_j\right)_\ell\right\}\right)$ be the set of auction bundles provided by seller s_j. Then ${}^{all}Ac_{all} = \biguplus_{i=1}^{c}{}^{all}Ac_i$ represents the set of all possible multiset

auction bundles requested by all consumers, and ${}^{all}As_{all} = \biguplus_{i=1}^{s} {}^{all}As_i$ represents the set of all possible multiset auction bundles provided by all sellers.

We assume that all sellers are willing to sell a bundle as long as it covers the cost of maintaining that bundle. Hence $Vs_j\left(As_j\right)_h$ may also be seen as the maintenance cost for seller s_j on the h^{th} auction bundle $\left(As_j\right)_h$, over the time period $\left[\tau_s, \tau_e\right]$. $\sum_{\ell=1}^{h} Vc_i\left(Ac_i\right)_\ell$ represents the sum of consumer c_i's valuation of all her auction bundles, and $\sum_{\ell=1}^{h} Vs_j\left(As_j\right)_\ell$ represents the sum of seller s_j's valuation of all her auction bundles. Therefore, $\sum_{i=1}^{c}\sum_{\ell=1}^{h} Vc_i\left(Ac_i\right)_\ell$ represents the sum of all consumers' valuations of all auction bundles, and $\sum_{j=1}^{s}\sum_{\ell=1}^{h} Vs_j\left(As_j\right)_\ell$ represents the sum of all sellers' valuations of all bundles. We define $contains_c\left(Ac, Ac^{'}\right)$ as True when:

$$\left[x\left(Ac\right) \supseteq x\left(Ac^{'}\right)\right] \wedge \left[\tau_s\left(Ac\right) \le \tau_s\left(Ac^{'}\right)\right] \wedge \left[\tau_e\left(Ac\right) \ge \tau_e\left(Ac^{'}\right)\right]$$

and $contains_s\left(As, As^{'}\right)$ is True when:

$$\left[x\left(As\right) \subseteq x\left(As^{'}\right)\right] \wedge \left[\tau_s\left(As\right) \ge \tau_s\left(As^{'}\right)\right] \wedge \left[\tau_e\left(As\right) \le \tau_e\left(As^{'}\right)\right]$$

Let consumer c's original requested auction bundle be denoted as Ac'. During the course of an auction, the consumer will only be willing to substitute Ac' with another auction bundle Ac if it meets either one of the two criteria:

$(i)\ Ac \equiv Ac' \qquad where\ Vc_i(Ac) = Vc_i(Ac')$

$(ii)\ contains_c(Ac, Ac') \qquad where\ Vc_i(Ac) = \begin{cases} \max\limits_{Ac' \in^{all} Ac_i : contains(Ac', Ac)} V(Ac') & if\ Ac'\ exists \\ 0 & otherwise \end{cases}$

Similarly, seller s will provide a substitute auction bundle if it meets either one of the two criteria:

$(i)\ As \equiv As' \qquad where\ Vs_i(As) = Vs_i(As')$

$(ii)\ contains_c(As, As') \qquad where\ Vs_i(As') = \begin{cases} \min\limits_{As' \in^{all} As_i : contains(As, As')} V(As') & if\ As'\ exists \\ \infty & otherwise \end{cases}$

Let $P(Z)$ denote trading price of a target auction bundle Z. For a trade to be possible between a seller and consumer, it is necessary for the trading price *P(Z)* to be in the range $\exists_{i,j} : Vs_j(Y) \le P(Z) \le Vc_i(X)$, where *Y, Z*, and *X* are auction bundles. Note that the auction resource bundles are required to meet the following requirements: $contains_s(Z, Y) \wedge contains_c(Z, X)$.

For example, if a consumer c requests for bundle X consisting of resources {*a,b,c*}, she will not be willing to accept auction bundles consisting of only resource bundle {*a,c*} since the required resource b is not in the resource bundle. She may, however, consider purchasing bundle *Z = {a, b, c, d}* if the following conditions are met:

$contains_c(Z, X)$

$P(Z) \le Vc(X)$

A feasible pair matching of auction bundles between seller s and consumer *c* is represented as: $(X_c, Y_s) : where\ X_c \in^{all} A_c, Y_s \in^{all} A_s, contains_c(Y_s, X_c), Vs(Y) \le Vc(X)$

Let the transaction T be represented by:

$$\left\{ \left(X_{k_1}, Y_{\ell_1}\right), \cdots, \left(X_{k_w}, Y_{\ell_w}\right) \;\middle|\; \begin{array}{c} X_{k_i} \in {}^{all}Ac_{all}, Y_{\ell_i} \in {}^{all}As_{all} \\ contains_c\left(X_{k_i}, Y_{\ell_i}\right) \\ \biguplus_{i=1}^{c}\left\{ {}^{c_i}X_{k_i} \right\} \subseteq {}^{all}Ac_{all},\ \biguplus_{j=1}^{s}\left\{ {}^{s_j}Y_{\ell_j} \right\} \subseteq {}^{all}As_{all} \\ Vs_j\left(Y_{\ell_i}\right) \le P(Z) \le Vc_i\left(X_{k_i}\right) \end{array} \right\}$$

We define $NonSell = \left\{ (i,j)_1, \cdots, (i,j)_k \right\}$ where each identified consumer and seller (i,j) pairs are not allowed to trade in any auctions. In some auctions, a seller in one ongoing auction may participate as a consumer in another auction. As such, *NonSell* is used to prevent an auctioneer from attempting to match the same seller with herself (as a consumer in another auction).

Under normal circumstances, a typical consumer's goal is always to maximize her individual utility $\arg\max_T \sum_{X_{k_j}: c\left(X_{k_j}\right)=c_i} Vc_i\left(X_{k_j}\right) - P\left(X_{k_j}\right)$.

The utility maximization for all consumers can be represented as:

$$\max_T \sum_c \sum_{X_{k_j}: c\left(X_{k_j}\right)=c_i} Vc_i\left(X_{k_j}\right) - P\left(X_{k_j}\right)$$

A seller's goal is always to maximize her individual profit where: $\arg\max_T \sum_{X_{k_j}: s\left(X_{k_j}\right)=s_i} P\left(X_{k_j}\right) - Vs_i\left(X_{k_j}\right)$.

The profit maximization for all sellers can be represented as:

$$\max_T \sum_s \sum_{X_{k_j}: s\left(X_{k_j}\right)=s_i} P\left(X_{k_j}\right) - Vs_i\left(X_{k_j}\right)$$

If the goal is to maximize global welfare, it is to achieve:

$$\arg\max_{T}\left[\sum_{\left(X_{k_i},X_{k_j},P\right)\in T,X_{k_j}:s\left(X_{k_j}\right)=s_j,X_{k_i}:c\left(X_{k_i}\right)=c_i}\left(\left(Vc_i\left(X_{k_i}\right)-P\left(X\right)\right)+\left(P\left(X\right)-Vs_i\left(X_{k_i}\right)\right)\right)\right]$$

Where the purchase price $\left(P\left(X\right)\right)$ is taken to be $P\left(X\right)=\left(\frac{P_c+P_s}{2}\right)$

8.2 One-shot Centralized Combinatorial Exchange (OCCE)

In an OCCE there are multiple consumers and multiple sellers participating in an auction, held by a central auctioneer. In this environment, we make the assumption that:

I. The central auctioneer has full knowledge of all seller's and consumer's bundle valuations through auction bids, and of all available resources in the environment.

II. The single central auctioneer is the sole authorized entity to hold auctions as well as to determine the outcome of the auction by matching consumers and sellers.

III. In cases where there is a bidding tie, the participant with earlier bid submission time wins the bid. If the submission is also a tie, the auctioneer randomly selects a winner with probability P_{random} from the bidders whose bids are in the tie. E.g. consumer c_1 submitted $Ac_1^{t=\tau_1}$, c_2 submitted $Ac_2^{t=\tau_2}$.

$$winner=\begin{Bmatrix} if\ \tau_1<\tau_2, c_1\ wins \\ else\, if\ \tau_1>\tau_2, c_2\ wins \\ else\, use\, P_{random} \end{Bmatrix}$$

IV. The central auctioneer concludes an auction, making auctioning decisions based on one of three goals:

   i. Seller's profit maximization (favors sellers)

ii. Consumer's purchase cost minimization (favors consumers)

iii. Global utility maximization (favors global welfare)

There are two types of models studied in the OCCE: a closed model (Section 8.2.1), and an open model (Section 8.2.2).

8.2.1. One-shot Centralized Closed CE (OCCCE)

In a system with a centralized auctioneer, both consumers and sellers have no knowledge of the ongoing bids since it is a closed auction. As such, each consumer submits auction bundle requests based on the jobs on hand. Based on the submitted auction bundle requests, each participating seller will submit auction bundle offers to the central auctioneer. It is the job of the central auctioneer to match the consumers and sellers together. The formalization follows that of the general formulations discussed earlier.

Example 1:

Let $X = \{ CPU_1, CPU_2, MEM_1, MEM_2 \}$ to represent the different types of unique resources available in market 1. The multiset representation of possible bundles is shown in Table 8.2 (where *k*=number of elements in a particular multiset, and *n*=number of distinct elements in the resource set *X*):

Table 8.2 Multiset Representation of Bundles in Market 1.

Multiset #	n=4, k=1	n=4, k=2	n=4, k=3	n=4, k=4
1	CPU_1	CPU_1CPU_1	$CPU_1CPU_1CPU_1$	$CPU_1CPU_1CPU_1CPU_1$
2	CPU_2	CPU_1CPU_2	$CPU_1CPU_1CPU_2$	$CPU_1CPU_1CPU_1CPU_2$
3	MEM_1	CPU_1MEM_1	$CPU_1CPU_1MEM_1$	$CPU_1CPU_1CPU_1MEM_1$
4	MEM_2	CPU_1MEM_2	$CPU_1CPU_1MEM_2$	$CPU_1CPU_1CPU_1MEM_2$
5		CPU_2CPU_2	$CPU_2CPU_2CPU_1$	$CPU_1CPU_1CPU_2CPU_1$
6		CPU_2MEM_1	$CPU_2CPU_2CPU_2$	$CPU_1CPU_1CPU_2CPU_2$
7		CPU_2MEM_2	$CPU_2CPU_2MEM_1$	$CPU_1CPU_1CPU_2MEM_1$
8		MEM_1MEM_1	$CPU_2CPU_2MEM_2$	$CPU_1CPU_1CPU_2MEM_2$
9		MEM_1MEM_2	$MEM_1MEM_1CPU_1$	$CPU_1CPU_1MEM_1CPU_1$
10		MEM_2MEM_2	$MEM_1MEM_1CPU_2$	$CPU_1CPU_1MEM_1CPU_2$
11			$MEM_1MEM_1MEM_1$	$CPU_1CPU_1MEM_1MEM_1$
12			$MEM_1MEM_1MEM_2$	$CPU_1CPU_1MEM_1MEM_2$
13			$MEM_2MEM_2CPU_1$	$CPU_1CPU_1MEM_2CPU_1$
14			$MEM_2MEM_2CPU_2$	$CPU_1CPU_1MEM_2CPU_2$
15			$MEM_2MEM_2MEM_1$	$CPU_1CPU_1MEM_2MEM_1$
16			$MEM_2MEM_2MEM_2$	$CPU_1CPU_1MEM_2MEM_2$
17			$CPU_1CPU_2MEM_1$	$CPU_1CPU_2CPU_2CPU_2$

Table 8.2 – Continued

18			CPU_1CPU_2MEM2	$CPU_1CPU_2CPU_2MEM_1$
19			$CPU_2MEM_1MEM_2$	$CPU_1CPU_2CPU_2MEM_2$
20			$MEM_1MEM_2CPU_1$	:
:				:
25				$CPU_2CPU_2MEM_2MEM_2$
:				:
:				:
35				:
	4 cases	10 cases	20 cases	35 cases

Table 8.3(a) represents units of different types of resource bundles requested by four different consumers and Table 8.3(b) represents the available resource bundles offered by three sellers in market 1. Table 8.3(c) lists the different types of resources available in the market.

Table 8.3 Market 1 Consumer's Requests.

Consumer	CPU_1	CPU_2	MEM_1	MEM_2
1	2		2	
2		1		1
3		2		1
4	1			2

Table 8.4 Market 1 Seller's Offerings.

Seller	CPU_1	CPU_2	MEM_1	MEM_2
1	2			2
2		4		4
3	2		2	

Table 8.5 Market 1 Resource Types.

Resources	
CPU_1	1 GHz, 1Xcore processor
CPU_2	2 GHz, 1Xcore processor
MEM_1	1 X 512 MB
MEM_2	1 X 1024 MB

Assume the consumers have the following valuation for their requested bundles:

$$C_1: \left.\begin{matrix} CPU_1 X\,2 \\ MEM_1 X\,2 \end{matrix}\right\} \begin{matrix} valuation\ \ Vc_1\left(x_{4,11}\right)=60 \\ \tau_{now,now+6hrs} \end{matrix}$$

$$C_2: \left.\begin{matrix} CPU_2 X\,1 \\ MEM_2 X\,1 \end{matrix}\right\} \begin{matrix} valuation\ \ Vc_2\left(x_{2,7}\right)=30 \\ \tau_{now,now+5hrs} \end{matrix}$$

$$C_3: \left.\begin{matrix} CPU_2 X\,2 \\ MEM_2 X\,1 \end{matrix}\right\} \begin{matrix} valuation\ \ Vc_3\left(x_{3,8}\right)=40 \\ \tau_{now,now+7hrs} \end{matrix}$$

$$C_4: \left.\begin{matrix} CPU_1 X\,1 \\ MEM_2 X\,2 \end{matrix}\right\} \begin{matrix} valuation\ \ Vc_4\left(x_{3,13}\right)=40 \\ \tau_{now,now+4hrs} \end{matrix}$$

Table 8.6 shows the available bundles and their respective valuations for each seller:

Table 8.6 Seller Bundles and Valuations.

S_1: CPU_1 X 2, MEM_2 X2

Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_1	10	CPU_1 CPU_1	20	CPU_1 CPU_1 MEM_2	30	CPU_1 CPU_1 MEM_2 MEM_2	40
MEM_2	10	CPU_1 MEM_2	20	CPU_1 MEM_2 MEM_2	30		
		MEM_2 MEM_2	20				

S_2: CPU_2 X 4, MEM_2 X 4

Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_2	10	CPU_2 CPU_2	20	CPU_2 CPU_2 CPU_2	30	CPU_2 CPU_2 CPU_2 CPU_2	40
MEM_2	10	CPU_2 MEM_2	25	CPU_2 CPU_2 MEM_2	30	CPU_2 CPU_2 CPU_2 MEM_2	45
		MEM_2 MEM_2	20	CPU_2 MEM_2 MEM_2	30	CPU_2 CPU_2 MEM_2 MEM_2	40
				MEM_2 MEM_2 MEM_2	30	CPU_2 MEM_2 MEM_2 MEM_2	40
						MEM_2 MEM_2 MEM_2 MEM_2	40

S_3: CPU_1 X 2, MEM_1 X2

Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_1	10	CPU_1 CPU_1	20	CPU_1 CPU_1 MEM_1	25	CPU_1 CPU_1 MEM_1 MEM_1	30
MEM_1	10	CPU_1 MEM_1	20	CPU_1 MEM_1 MEM_1	30		
		MEM_1 MEM_1	20				

After analyzing both consumers' requests and sellers' offerings, the auctioneer might yield an assignment as shown in Table 8.7 (depicted in Figure 8.4).

Table 8.7 Resource Assignment for Example 1.

S_1: CPU_1 X 2, MEM_2 X2

Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_1	10	CPU_1 CPU_1	20	CPU_1 CPU_1 MEM_2	30	CPU_1 CPU_1 MEM_2 MEM_2	40
MEM_2	10	CPU_1 MEM_2	20	***CPU_1 MEM_2 MEM_2***	***30***		
		MEM_2 MEM_2	20				

S_2: CPU_2 X 4, MEM_2 X 4

Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_2	10	CPU_2 CPU_2	20	CPU_2 CPU_2 CPU_2	30	CPU_2 CPU_2 CPU_2 CPU_2	40
MEM_2	10	***CPU_2 MEM_2***	***25***	***CPU_2 CPU_2 MEM_2***	***30***	CPU_2 CPU_2 CPU_2 MEM_2	45
		MEM_2 MEM_2	20	CPU_2 MEM_2 MEM_2	30	CPU_2 CPU_2 MEM_2 MEM_2	40
				MEM_2 MEM_2 MEM_2	30	CPU_2 MEM_2 MEM_2 MEM_2	40
						MEM_2 MEM_2 MEM_2 MEM_2	40

S_3: CPU_1 X 2, MEM_1 X2

Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_1	10	CPU_1 CPU_1	20	CPU_1 CPU_1 MEM_1	25	***CPU_1 CPU_1 MEM_1 MEM_1***	***30***
MEM_1	10	CPU_1 MEM_1	20	CPU_1 MEM_1 MEM_1	30		
		MEM_1 MEM_1	20				

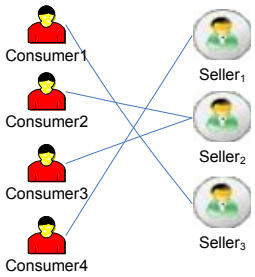

Figure 8.4 Auctioneer Decision Outcome.

Example 2:

Assuming a similar scenario as in example 1 with the addition of a fifth consumer with the following resource request:

$$C_5 : \begin{matrix} CPU_2 X\,2 \\ MEM_2 X\,2 \end{matrix} \Big\} \begin{matrix} valuation\ Vc_5\left(x_{4,25}\right) = 55 \\ \tau_{now,now+5hrs} \end{matrix}$$

Table 8.8 Consumer and Seller Valuations for Example 2.

Consumer's valuation	Seller's valuation (S_2)	Profit
C2: 30	25	5
C3: 40	30	***10***
C5: 55	45	***15***

Everything would remain the same except for seller $\left(S_2\right)$'s resource sales:

Table 8.9 Resource Assignment for Example 2.

S_2: CPU_2 X 4, MEM_2 X 4							
Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_2	10	CPU_2 CPU_2	20	CPU_2 CPU_2 CPU_2	30	CPU_2 CPU_2 CPU_2 CPU_2	40
MEM_2	10	***CPU_2 MEM_2***	***25***	***CPU_2 CPU_2 MEM_2***	***30***	CPU_2 CPU_2 CPU_2 MEM_2	45
		MEM_2 MEM_2	20	CPU_2 MEM_2 MEM_2	30	***CPU_2 CPU_2 MEM_2 MEM_2***	***40***
				MEM_2 MEM_2 MEM_2	30	CPU_2 MEM_2 MEM_2 MEM_2	40
						MEM_2 MEM_2 MEM_2 MEM_2	40

Example 3:

Assuming a similar scenario as in example 1, with the addition of a fifth consumer with a different resource request from that of example 2:

$$C_5 : \left.\begin{matrix} CPU_2 X 3 \\ MEM_2 X 1 \end{matrix}\right\} \begin{matrix} valuation\ Vc_5\left(x_{4,30}\right) = 35 \\ \tau_{now,now+5hrs} \end{matrix}$$

Table 8.10 Consumer and Seller Valuations for Example 3.

Consumer's valuation	Seller's valuation (S_2)	Profit
C2: 30	25	***5***
C3: 40	30	10
C5: 55	45	***10***

Although selling $\left(x_{3,8}\right)$ to C_3 would seem more profitable for Seller $\left(S_2\right)$ than selling $\left(x_{2,7}\right)$ to C_2, seller $\left(S_2\right)$ would end up selling fewer bundles, since he would not be able to sell $\left(x_{4,30}\right)$ to C_5 as a result. Hence a better allocation would result in:

Table 8.11 Resource Assignment for Example 3.

S_2: CPU_2 X 4, MEM_2 X 4							
Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation	Resource bundle	Bundle valuation
CPU_2	10	CPU_2 CPU_2	20	CPU_2 CPU_2 CPU_2	30	CPU_2 CPU_2 CPU_2 CPU_2	40
MEM_2	10	***CPU_2 MEM_2***	***25***	***CPU_2 CPU_2 MEM_2***	***30***	***CPU_2 CPU_2 CPU_2 MEM_2***	***45***
		MEM_2 MEM_2	20	CPU_2 MEM_2 MEM_2	30	CPU_2 CPU_2 MEM_2 MEM_2	40
				MEM_2 MEM_2 MEM_2	30	CPU_2 MEM_2 MEM_2 MEM_2	40
						MEM_2 MEM_2 MEM_2 MEM_2	40

8.2.2. One-shot Centralized Open CE (OCOCE)

In an open OCOCE model, the auction essentially becomes a true ascending auction model for the consumers as competing consumers have full knowledge of what competing consumers are bidding on and which auction bundles are being offered by all sellers. Therefore, participating consumers with overlapping resource bundle requirements may end up bidding for similar auction bundles if the supply of resource auction bundles is limited. Such consumers may have to continuously increase their bids in an attempt to outbid their competitors to the point where the bids placed are exactly their valuations for a bundle [32]. Similarly, the same environment becomes a descending auction model for the sellers who would continuously decrease their bids so long as it remains more than or equal to their valuation for the bundle.

The formulation is similar to that of a One-shot Centralized Closed Combinatorial Exchange (OCCCE) until consumers are bidding on the same bundle in the open market where they have to start competing against their competitors in an attempt to win the auction. The bidding becomes an ascending auction when the following condition is satisfied: $\exists_{i,j;i\neq j}\begin{Bmatrix} X \in {}^{all}Ac_i \wedge Y \in {}^{all}Ac_j \wedge Z \in {}^{all}As \\ Vc_i(X) \geq P(Z), Vc_j(Y) \geq P(Z), Vs(Z) \leq P(Z) \\ contains(z,x) \wedge contains(z,y) \end{Bmatrix}$ where *Z* is an auction bundle offered by seller *s*, and *X* and *Y* are auction bundles requested by both consumers $c_i \; and \; c_j$ respectively. If Z satisfies both consumer c_i 's request for *X* and consumer c_j 's request for *Y*, and *Z* is the lowest priced currently available auction bundle offer, where: $Z = \underset{U \in {}^{all}As:contains_c(Z,X)}{\arg\min} Vc_i(U)$ and $Z = \underset{U \in {}^{all}As:contains_c(Z,Y)}{\arg\min} Vc_j(U)$, both consumers $c_i \; and \; c_j$ will be engaged in an ascending multi-round bidding for bundle *Z*. The winning transaction will be represented as: $\nexists(T_1 \cdots T_c), T_i = \arg\max_T Uc_i(T), \forall_{(X_i,Y_i)\in T_i}, \forall_{(X_j,Y_j)\in T_j}, Y_i \neq Y_j$.

Similarly, if sellers offer auction bundles satisfying the same consumer's auction bundle request in an open market where the following condition is satisfied:

$$\exists_{i,j;i\neq j}\begin{Bmatrix} X \in {}^{all}As_i \wedge Y \in {}^{all}As_j \wedge Z \in {}^{all}Ac \\ Vs_i(X) \leq P(Z), Vs_j(Y) \leq P(Z), Vc(Z) \geq P(Z) \end{Bmatrix}$$

Where *Z* is an auction bundle requested by consumer *c*, and *X* and *Y* are auction bundles offered by both sellers $s_i \; and \; s_j$ respectively. If *Z* satisfies both seller s_i 's offer for *X* and seller s_j 's offer for *Y*, and *Z* is the highest priced currently available auction bundle request, where: $Z = \underset{U \in {}^{all}Ac:contains_s(Z,X)}{\arg\max} Vs_i(U)$ and $Z = \underset{U \in {}^{all}Ac:contains_s(Z,Y)}{\arg\max} Vs_j(U)$, both sellers $s_i \; and \; s_j$ would be competing in a descending auction for bundle *Z*. The winning transaction will be represented

as: $\nexists\left(T_1 \cdots T_s\right), T_i = \arg\min_T Us_i\left(T\right), \forall_{\left(X_i, Y_i\right) \in T_i}, \forall_{\left(X_j, Y_j\right) \in T_j}, Y_i \neq Y_j$. After the winners for all the competing auction bundles (sellers and consumers) have been determined, the remaining formalization would again be similar to that for a One-shot Centralized Closed Combinatorial Exchange (OCCCE) model. The central auctioneer will match the remaining non-clashing auction bundles to the respective consumers and sellers.

8.3 One-shot Decentralized Combinatorial Exchange (ODCE)

A DCE environment contains two or more auction markets within the grid environment. There are several ways the creation of separate auction markets may be determined, such as participant locality, similarities in job types, datasets locality, different virtual organizations within a grid environment etc. If there is no participation overlapping in the DCE environment, each seller/consumer participate in the auction through one (and only one) of the auctioneers. In other words, each auctioneer only serves a subset of all bidders. On the other hand, if participation overlapping is allowed, each seller/consumer may participate in the auction through one or more auctioneers. Section 8.3.1 discusses cases with no participation overlapping, and Section 8.3.2 discusses on cases where participation overlapping is allowed.

8.3.1. One-shot Centralized Closed CE (ODCCE) with no overlapping

An ODCCE with no overlapping is very much similar to a One-shot Centralized Closed Combinatorial Exchange (OCCCE) albeit with multiple micro instances where there are multiple consumers and multiple sellers participating in each auction instance held by a central auctioneer instance. In this environment, we make the assumption that:

I. The central auctioneer in each instance has full knowledge of all seller's and consumer's bundle valuations within the same instance without knowledge of all seller's and consumer's bundle valuations in other instances. Similarly for its knowledge of all available resources in the environment.

II. The central auctioneer instance is the sole authorized auctioneer to hold auctions for all sellers and consumers in the market within each instance.

III. The central auctioneer instance concludes an auction in a single round, making auctioning decisions based on three goals:

i. Seller's profit maximization (favors sellers)

ii. Consumer's purchase cost minimization (favors consumers)

iii. Global utility maximization (favors global welfare)

Since the formulation for One-shot Decentralized Closed CE (ODCCE) with no overlapping is similar to that of multiple instances of a One-shot Centralized Closed CE (OCCCE) model, readers are referred to Section 8.2.1 for formalization representations. Figure 8.5 shows an example of an ODCCE model with no overlapping.

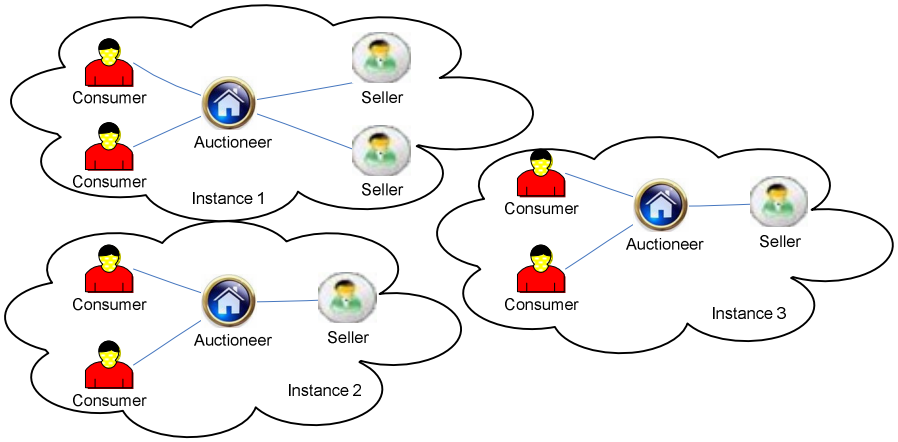

Figure 8.5 Decentralized Closed CE With No Overlapping.

8.3.2. One-shot Centralized Closed CE (ODCCE) with overlapping

In a One-shot Decentralized Closed CE (ODCCE) with overlapping auctioning model, consumers and sellers have the option of submitting auction bundle requests and offers to multiple auctioneers. The decision of whether to submit multiple auction bundle requests or offers is affected by respective aggressiveness indices (discussed in Sections 6.4.1.12 and 6.4.2.6). Figure 8.6 shows an example of a Decentralized Combinatorial Exchange (DCE) model with overlapping.

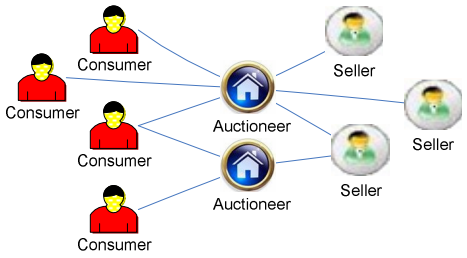

Figure 8.6 Decentralized CE With Overlapping.

By submitting bids to multiple auctioneers, the consumer has the opportunity to take part in multiple auctions, hence increasing her chances of winning in an auction. However, she also runs the risk of possibly winning more than one similar auction bundle. Overlapping bids may occur when a consumer has a high priority job on hand or when a job is quickly approaching its deadline. When this happens, a consumer may attempt to raise her chances of acquiring that urgently needed resource bundle by placing bids with multiple auctioneers. Note that this is different from submitting a bid request for multiple auction bundles to a single auctioneer as that auctioneer may not assign that multiple-unit bundle to that consumer unless she wins the auction. Similarly, sellers may 'oversell' their resources by placing bundle offers to multiple auctioneers in an attempt to raise the probability of selling their resources. Sellers may 'oversell' for the following reasons:

I. Since resources unused would still incur maintenance cost, sellers may try to promote their resource bundle offerings by sending auction bundle offers to multiple auctioneers so as to reach a larger consumer market.

II. This may be used as a profit maximizing strategy by sellers, which is what many of the airline companies practice.

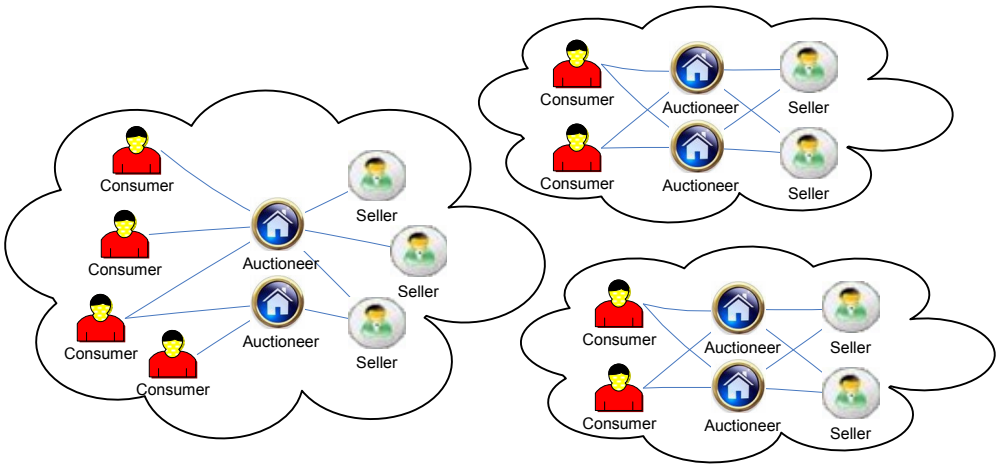

Figure 8.7 Multiple Decentralized CE With Overlapping.

However, in addition to the extra auction participation cost incurred, auction participants who engage in either 'overbuying' or 'overselling' also run a possible cancellation penalty fee if they win the auction from multiple auctioneers and had to cancel the extra winning bid because they cannot purchase/provide all bundles as a result.

For example, if consumer c_i submitted $Ac_i^{\tau} = \left\{ {}^{A}X_{k,n}, {}^{A}\tau_s, {}^{A}\tau_e, {}^{A}V \right\}$ as auction bid with two different auctioneers and resulted in winning both auctions: consumer ^{c_i} is matched with $\left(Ac_i, Ys_j \right)$ and $\left(Ac_i, Ys_k \right)$, where $contains_c \left(Ys_j, Ac_i \right) and\ contains_c \left(Ys_k, Ac_i \right)$, $Ys_j = \left\{ us_j^t, {}^{Y}X_{k,n}, {}^{Y}\tau_s, {}^{Y}\tau_e, {}^{Y}V \right\}$ and $Ys_k = \left\{ us_k^t, {}^{Y}X_{k,n}, {}^{Y}\tau_s, {}^{Y}\tau_e, {}^{Y}V \right\}$. Consumer c_i has to determine which winning auction bundle to drop by calculating the one bundle yielding the lowest utility:

$$\min\left(\left[Vc_i\left(Ac_i\right)-\left(P\left(Ys_j\right)+\pi_{c_i}^{s_j}\right)\right],\left[Vc_i\left(Ac_i\right)-\left(P\left(Ys_k\right)+\pi_{c_i}^{s_k}\right)\right]\right)$$ where $\pi_{c_i}^{s_j}$ denotes the penalty fee consumer i pays to seller j. After the minimum utility yielding auction has been determined, consumer c_i pays the canceling penalty to the seller whose lowest utility yielding

auction bundle has been returned. Assuming $seller\ j$'s auction bundle has been canceled. The penalty payment to $seller_j$ is calculated as:

$$c_\pi = max\left(\left(\frac{1}{us_j} * P\left(Ys_j\right)\right), x * P\left(Ys_j\right)\right)$$

For details on cancellation fee for both consumers and sellers, readers are referred to Sections 6.4.1.10 and 6.4.2.5, respectively.

8.3.3. One-shot Decentralized Open CE (ODOCE) with no overlapping

A ODOCE model without overlapping has characteristics much like that of multiple smaller One-shot Centralized Open CE models where the open auctioning system transforms the auctioning market into an ascending open bidding system for the consumers while sellers would be engaged in a descending open bidding auction.

For each market instance, consumer bidding becomes an ascending auction when the following condition is satisfied:

$$\exists_{i,j;i\neq j}\begin{Bmatrix} X \in {}^{all}Ac_i \wedge Y \in {}^{all}Ac_j \wedge Z \in {}^{all}As \\ Vc_i\left(X\right) \geq P\left(Z\right), Vc_j\left(Y\right) \geq P\left(Z\right), Vs\left(Z\right) \leq P\left(Z\right) \end{Bmatrix}$$ where *Z* is an auction bundle offered by seller *s*, and *X* and Y are auction bundles requested by both consumers $c_i\ and\ c_j$ respectively. If *Z* satisfies both consumer c_i's request for *X* and consumer c_j's request for Y, and *Z* is the lowest priced currently available auction bundle offer, where: $Z = \underset{U \in {}^{all}As:contains_c(Z,X)}{\arg\min} Vc_i\left(U\right)$ and $Z = \underset{U \in {}^{all}As:contains_c(Z,Y)}{\arg\min} Vc_j\left(U\right)$, both consumers $c_i\ and\ c_j$ will be engaged in an ascending multi-round bidding for bundle Z. The winning transaction will be represented as: $\nexists\left(T_1 \cdots T_c\right), T_i = \underset{T}{\arg\max}\, Uc_i\left(T\right), \forall_{(X_i,Y_i)\in T_i}, \forall_{(X_j,Y_j)\in T_j}, Y_i \neq Y_j$.

Similarly, if sellers become engaged in a descending auction if they offer auction bundles satisfying the same consumer's auction bundle request in an open market where the following condition is satisfied:

$$\exists_{i,j;i\neq j}\begin{Bmatrix} X \in {}^{all}As_i \wedge Y \in {}^{all}As_j \wedge Z \in {}^{all}Ac \\ Vs_i(X) \leq P(Z), Vs_j(Y) \leq P(Z), Vc(Z) \geq P(Z) \end{Bmatrix}$$

where *Z* is an auction bundle requested by consumer *c*, and *X* and *Y* are auction bundles offered by both sellers $s_i \; and \; s_j$ respectively. If *Z* satisfies both seller s_i 's offer for *X* and seller s_j 's offer for *Y*, and *Z* is the highest priced currently available auction bundle request, where: $Z = \underset{U \in {}^{all}Ac:contains_s(Z,X)}{\arg\max} Vs_i(U)$ and $Z = \underset{U \in {}^{all}Ac:contains_s(Z,Y)}{\arg\max} Vs_j(U)$, both sellers $s_i \; and \; s_j$ would be competing in a descending auction for bundle *Z*. The winning transaction will be represented as: $\nexists(T_1 \cdots T_s), T_i = \underset{T}{\arg\min} Us_i(T), \forall_{(X_i,Y_i)\in T_i}, \forall_{(X_j,Y_j)\in T_j}, Y_i \neq Y_j$. Similar to the One-shot Centralized Open CE model, the auctioneer within each market instance will determine the matching of the remaining non-clashing auction bundles to the respective consumers and sellers.

8.3.3. One-shot Decentralized Open CE (ODOCE) with overlapping

Like ODCCE with overlapping, consumers and sellers in a ODOCE with overlapping model may participate with multiple auctioneers in hope of attaining higher chances of securing auction bundles at the extra cost of possible penalty fees on top of the added auction participation charges. However, with the added benefit of having full knowledge of all ongoing bids placed by all participants in an open market, a participant could avoid possibly paying any penalty fees by intentionally bidding on auction bundles which already have a bidder, and backing out of all ascending auctions (by stopping bidding and letting the competitor win) when the desired auction bundle has been won in one of the other ongoing ascending bid CE auctions. However, since the consumer participating with multiple auctioneers incurs a higher cost when compared to consumers who only place bids with a single auctioneer, her only chance of winning in any one of the auctions is when her competing bidders have a much lower valuation for the same auction bundle.

For example, consumer c_i submitted $Ac_i^{\tau} = \left\{ {}^{A}X_{k,n}, {}^{A}\tau_s, {}^{A}\tau_e, {}^{A}V \right\}$ as auction bid with three different auctioneers from three different auction markets {*1, 2, 3*}. Consumer c_i competes with one other local consumer $\left\{ c_{1_{market1}}, c_{1_{market2}}, c_{1_{market3}} \right\}$ in all of the three different markets. In market1, market2, and market3:

$$\exists_{i, j_{market\_x}; i \neq j_{market\_x}} \left\{ \begin{array}{l} X \in {}^{all}Ac_i \wedge Y \in {}^{all}Ac_{j_{market\_x}} \wedge Z \in {}^{all}As \\ Vc_i(X) \geq P(Z), Vc_{j_{market\_x}}(Y) \geq P(Z), Vs(Z) \leq P(Z) \end{array} \right\}$$ where *Z* is an auction bundle offered by seller *s*, and *X* and *Y* are auction bundles requested by both consumers $c_i \; and \; c_{j_{market\_x}}$ respectively. If *Z* satisfies both consumer c_i 's request for *X* and consumer $c_{j_{market\_x}}$'s request for Y, and Z is the lowest priced currently available auction bundle offer in market *x*, where:

$$Z = \underset{U \in {}^{all}As: contains_c(Z,X)}{\arg\min} Vc_i(U) \text{ and } \quad Z = \underset{U \in {}^{all}As: contains_c(Z,Y)}{\arg\min} Vc_{j_{market\_x}}(U), \text{ then both}$$

consumers $c_i \; and \; c_{market\_x}$ will be engaged in ascending multi-round bidding for bundle *Z*. The winning transaction will be the consumer with the highest bidding price (valuation) for the auction bundle:

$$\nexists (T_1 \cdots T_c), T_i = \arg\max_{T} Uc_i(T), \forall_{(X_i, Y_i) \in T_i}, \forall_{\left(X_{j_{market\_x}}, Y_{j_{market\_x}}\right) \in T_{j_{market\_x}}}, Y_i \neq Y_{j_{market\_x}}.$$

Similarly, sellers could also intentionally participate in auctions from multiple market instances where there exist other sellers bidding on the same auction bundle. When the seller has won in any of the descending bid auction, she will back out of the other auctions to avoid paying any penalty fees.

$$\exists_{i, j_{market\_x}; i \neq j_{market\_x}} \left\{ \begin{array}{l} X \in {}^{all}As_i \wedge Y \in {}^{all}As_{j_{market\_x}} \wedge Z \in {}^{all}Ac \\ Vs_i(X) \leq P(Z), Vs_{j_{market\_x}}(Y) \leq P(Z), Vc(Z) \geq P(Z) \end{array} \right\}$$

where *Z* is an auction bundle requested by consumer *c*, and *X* and *Y* are auction bundles offered by both sellers $s_i \; and \; s_{j_{market\_x}}$ respectively. If *Z* satisfies both seller s_i 's offer for *X* and seller $s_{j_{market\_x}}$'s offer for *Y*, and *Z* is the highest priced currently available auction bundle request, where: $Z = \underset{U \in {}^{all} Ac: contains_s(Z,X)}{\arg\max} Vs_i(U)$ and $Z = \underset{U \in {}^{all} Ac: contains_s(Z,Y)}{\arg\max} Vs_{j_{market\_x}}(U)$, then If both sellers $s_i \; and \; s_{j_{market\_x}}$ decided to choose to sell their respective auction bundles using this method, the model would turn into a conventional open descending first price CE auction for auction bundle *Z*. The winning transaction will be represented as: $\nexists (T_1 \cdots T_s), T_i = \underset{T}{\arg\min} Us_i(T), \forall_{(X_i,Y_i) \in T_i}, \forall_{\left(X_{j_{market\_x}}, Y_{j_{market\_x}}\right) \in T_{j_{market\_x}}}, Y_i \neq Y_{j_{market\_x}}$.

8.4 Repeated Centralized Combinatorial Exchange (RCCE)

In a repeated auction model, participants may choose their bidding strategies (discussed in detail in Chapter 9) based on prior historical information. Thus more auction participation experience may aid in better chances of winning in future auctions. However, since every participating bidder's experience depends on the type of previous jobs they had (since the type of job determines the type of auction bundles they will bid for), they tend to accumulate unique historical data over time. The usefulness of each participant's historical data depends on several factors such as bidding strategy adopted, consistency of bidder and that of other competitors, outcome from previous bidding strategies, bidding aggressiveness, etc. For example, a participant who has been losing all previous auctions probably has a very good idea of which bidding strategies do not work. But this information may not necessarily prove to be useful information for making future bids.

Consumers may collect historical information on:

I. Job type information (most recent m prices, averaged market prices)
II. Resource bundles (tracking demand for specific resource bundles)
III. Sellers' bundle valuations (based on when sellers accept or drop out of auctions)

IV. Competing consumers' bundle valuations (based on when consumers accept or drop out of auctions)
V. Best bidding strategies
    i. Max utility (maximize returns from each transaction)
    ii. Min cost (minimizing cost of purchasing auction bundles)
VI. Productivity (information on own performance)
VII. Cost (information on past costs incurred)
VIII. Overbuying experiences

Similarly, sellers may collect historical information:

I. Competing sellers' bundle valuations (based on when sellers accept or drop out of auctions)
II. Consumers' bundle valuations (based on when consumers accept or drop out of auctions)
III. Information on profits versus aggressiveness relationship
IV. Best bidding strategies
    i. Max profit/wealth
    ii. Min maintenance cost
    iii. Min resource bundle waste due to time slot fragmentation
    iv. Overselling experiences

8.4.1. Repeated Centralized Closed CE (RCCCE) with overlapping

When compared to their One-shot auction counterparts, the Repeated Centralized Closed CE model is quite similar to them except for the additional historical information from past participation. The added historical information arms the auction participants with better knowledge and understanding of the auctioning market. However, since this is a closed market model, useful historical information is limited to private information such as which bidding

strategy performed the best among all past adopted strategies, auction bundle estimates on sellers with whom the consumer has dealt with in the past and vice versa.

8.4.2. Repeated Centralized Open CE (RCOCE)

In a RCOCE model, each participant is well informed of all historical information of all other participants. Valuations of competing participants can easily be estimated by observing their trading behavior, as well as strategies adopted by each. As such, every participant is provided with a vast amount of information within the entire system where each participant can constantly improve themselves by learning from one another. However, this is at the very high cost of communications due to the movement of all trading information in a system with an open market.

8.5 Repeated Decentralized Combinatorial Exchange (RDCE)

Like Repeated Centralized CE (RCCE), the RDCE is also similar to their One-shot auction counterparts, but with the added historical information from past participation.

8.5.1. Repeated Decentralized Closed CE (RDCCE) with no overlapping

A RDCCE with no overlapping has similar characteristics as a One-shot Decentralized Closed CE (ODCCE) with no overlapping: each decentralized market instance acts as a standalone micro market but with the additional private historical information as well as with information on other participants with whom a participant has dealt with in the past.

8.5.2. Repeated Decentralized Closed CE (RDCCE) with overlapping

A RDCCE with overlapping provides more historical information than that of one without overlapping support, especially for participants who had attempted overlapping bids in the past. It provides useful private information which allows for fine tuning of respective over-buying/over-selling strategies adopted in the past. For instance, a seller who has not been able to sell her resources in the currently participating decentralized system may choose to withdraw from a worse performing market and participate in other decentralized markets by taking advantage of the overlapping markets.

8.5.3. Repeated Decentralized Open CE (RDOCE) with no overlapping

A RDOCE with no overlapping provides each participant with historical information on all participants within the same market instance. As such, each participating consumer (or seller) can better gauge other competitors' valuation as well as their bidding habits based on past historical information. This has the advantage of offering participants a tool to fine tune their respective bidding strategies over time. But since overlapping is not supported in this model, no participant has any knowledge of anything outside of their micro market instance.

8.5.4. Repeated Decentralized Open CE (RDOCE) with overlapping

In a RDOCE model with overlapping, consumers behave rather similar to that of a One-shot Decentralized Open CE with the added benefit of historical information - sellers will be able to take advantage of the overlapping to extend their reach to other decentralized communities. For example, if there is a strong demand from consumers for a particular auction bundle which cannot be met within the localized circle, one of the seller's strategies may be to cross boundaries and purchase the expected high-demand auction bundle (as a consumer) from sellers in another community and turn around to sell it to consumers in the originating auction boundary. This helps promoting sellers' bundle resources across different boundaries, and hence improving the overall resource allocation efficiency. Figure 8.8 shows a possible configuration of a complex DCE with overlapping. This provides a tradeoff between Repeated Centralized Open CE (RCOCE) and an isolated instance that the Repeated Decentralized Open CE (RDOCE) with no overlapping supports.

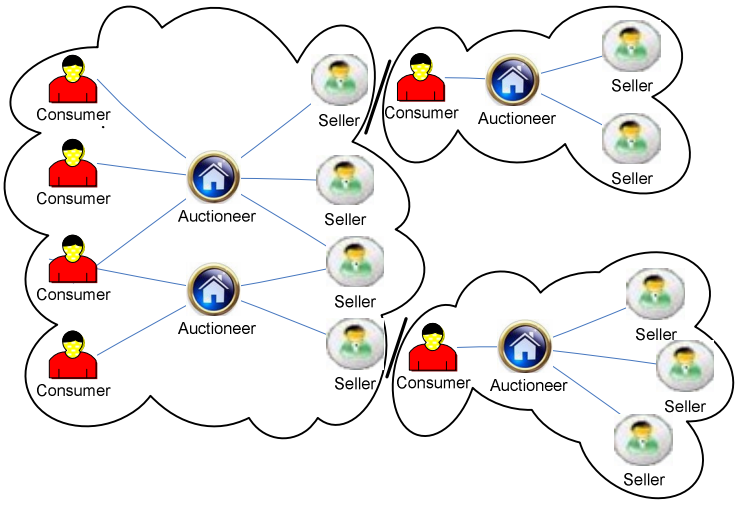

Figure 8.8 Decentralized CE With Overlapping.

CHAPTER 9

IMPLEMENTATION OF GRID MARKET MECHANISM

The implementation of auction based Grid market mechanisms will be discussed in this chapter. Section 9.1 describes the studies performed using brokers with job advertisements. Section 9.2 discusses the implementation of Grid market mechanisms using the combinatorial exchange methodology.

9.1 Auction Based Grid Resource Scheduling Using Brokers and Job Advertisements

In this work, performance and behavioral aspects of applying auctioning mechanisms using brokers with job advertisements is studied and analyzed.

9.1.1. Introduction

A computational cluster is a group of networked computers usually created by organizations for processing large data or computation intensive jobs. Several such (remote) clusters may be integrated together to form a computational grid. Thus, in a computational grid environment where data and computation intensive jobs are to be processed, there exists a vast collection of resources ready to process jobs of (virtual) organizations. Scheduling and resource management in a computational grid environment has been an area of extensive research due to its importance and complexity. Resources within a computational grid are likely to be heterogeneous, i.e., individual clusters or computers may be of different architectures, may be controlled by different operating systems and have diverse libraries. This often requires some form of resource matching such as [39] to effectively map jobs to resources. However, due to fluctuating demands, resource availability may rapidly change. In addition, scheduling and management policies of clusters are unlikely to be uniform over all clusters in the grid. With ubiquitous resources distributed throughout clusters, it is important to be able to effectively manage these resources as well as the assign jobs to take advantage of the available subset of

resources. Without effective monitoring and management of resources, information on resource availability and the condition and duration of such availability is unknown. Furthermore, efficient scheduling is necessary in order to keep a particular resource from being overwhelmed when a similar resource may be “sitting” at another location idling.

Scheduling within a computational grid environment is often concerned with the welfare of the resources or components as a whole as well as the well being of individuals. By welfare of the resources or components, we often refer to such issues as fairness, e.g., equal opportunities to use resources, distribution of wealth throughout the grid and the overall performance of all the virtual organizations combined as a whole. Individual well being, on the other hand, is often concerned with the maximization of satisfaction derived from participating in work related activities. Such derived satisfaction may include minimization of response time and cost while maximizing profit, throughput, and yield of earliest results. Thus, it is often necessary to be able to strike a balance between two contrasting goals: individual (per cluster) goals and system level utilization goals.

Individual (or local) goals, as the name implies, are more often concerned with maximizing the benefits individual entities can attain with minimal regards for the welfare of the rest of the system. On the other hand, system level goals are usually more concerned with getting the most out of the currently available resources globally, even if it is at the cost of sacrificing a small population in order to benefit the system utility as a whole. For example, in order to maximize system resource utilization, a system level goal could be to keep all resources busy with a minimal number of unassigned resources idling. An individual goal, however, may be to have some idling resources available so as to have instant access to those resources when need arises. Similarly, it is a system level goal to ensure fair distribution of wealth (when completion of jobs are rewarded by some means) while an individual is more often interested in maximizing local profits. As a result, it is a challenging task to satisfy individuals while still achieving system level goals at the same time.

To address the problem of resource management and job scheduling in a large, geographically distributed network of virtual organizations while observing the goals of cost minimization, improving utilization and efficiency, we propose a pull-based grid scheduling methodology which adopts the use of brokers with job advertisement and propagation within a grid environment. The main motivation for this scheme is to create an automated two tiered scheduling methodology to perform the tedious task of performing service discovery and task scheduling at the global level, while performing resource monitoring, utilization and efficiency control at the local level. To achieve the best attainable optimization at any point in time, participating sites are to remain motivated to offer their best services based on the job submitting host's preferred optimization settings. Global scheduling of jobs will be done at the broker level via a bidding process. The submitting host will have the privilege to choose the best available offer to suit its requirements. A pricing scheme is implemented as a trading mechanism in exchange for the services provided. This pricing mechanism will hence serve as a motivation for participating sites to compete for jobs so as to increase its overall wealth. As such, competing sites will be required to constantly monitor and improve their own resources, their utilization, and their efficiency to remain competitive.

The main goal of this work [41] is to perform analysis of the behavioral and performance aspects of two conventional schemes and the proposed bidding scheme while enforcing self motivated resource management technique with a combination of job optimization schemes (time and cost), along with support for re-negotiations in cases where matching fails. We will also look into the factors enticing both the participating entities to continue trading within a grid environment and the effects of implementing a penalty scheme.

9.1.2. Broker and Bid Based Scheduling

In our grid model, each cluster is associated with a head-node (gateway) running a software broker and other management software. An organization may have several head-nodes arranged into a meta-cluster, where the broker from one of the head-nodes is the broker

of the meta-cluster (the set of all meta-clusters forms a grid). In this grid environment there are resource consumers (entities that submit jobs) and resource providers (entities that service job requests). Brokers of resource consumers post jobs; brokers of resource providers bid on these jobs by posting their fees and estimated job completion time. The posting broker evaluates bids and selects among them.

The use of brokers alleviates the mundane and tedious tasks of resource monitoring and decision making from any single centralized entity within the grid environment. The automated brokers are responsible for submitting job advertisements, initiating and maintaining the job bidding process, participating in job bids, and performing re-negotiations should the need arise. A broker may play the role of a consumer (request broker), a provider (tender broker), or both at any point in time.

If a site has jobs to be executed, a job advertisement object consisting of job descriptions, job type definitions and constraints, is created and passed to its gateway broker. A job placement may require one of two types of optimization, namely time or cost optimization. A time optimized job places more emphasis on completing the job at the earliest possible time, while not exceeding its available budget set aside for the job (best effort time but limited cost). A cost optimized job, on the other hand, emphasizes finding a cheapest tender for processing the job while still completing the job by its specified deadline (limited turnaround time but best effort cost).

After receiving the job advertisement, the gateway requesting broker places the job advertisement which propagates to all of its immediate one-hop neighbors as well as to the cluster brokers within the same local site. (Figure 9.1 shows the inner workings of a requesting broker, while Figure 9.2 depicts that of a tender broker). At the same time, a bid expiration timeout event is placed (Figure 9.1, step 3) to signify the auction closing time for that job ID.

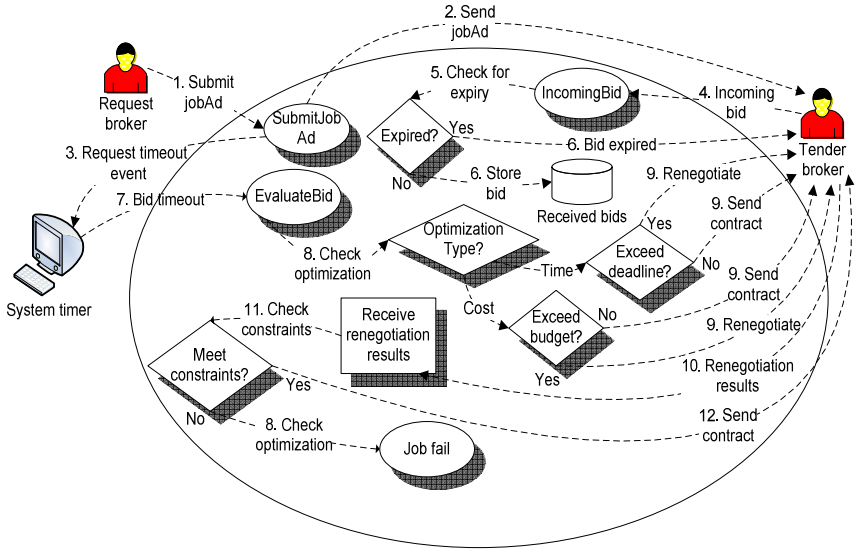

Figure 9.1 Request Broker.

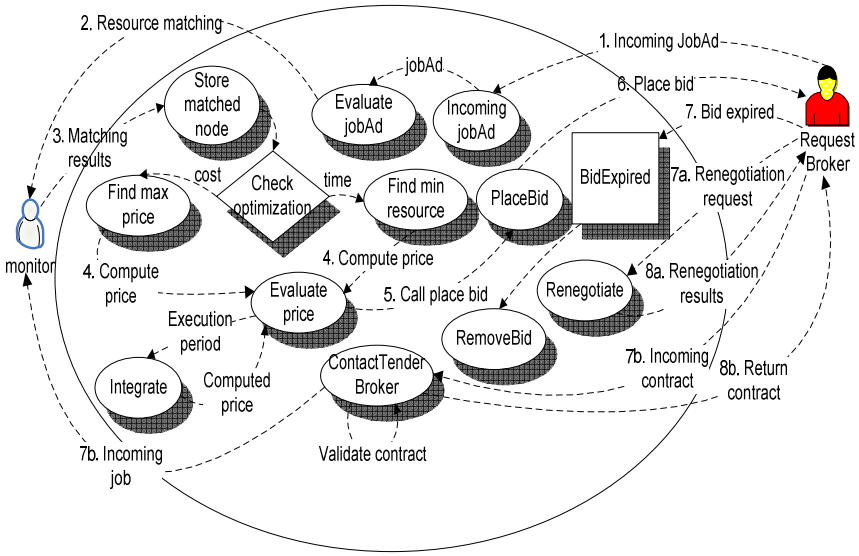

Figure 9.2 Tender Broker.

A tender broker receiving advertisements will perform a query of its local cluster monitors to search for nodes meeting the job requirements and constraints (Figure 9.2, step 2). The tender broker will then perform an analysis step to determine which node to engage in the bid submission (attempting to maximize profits while keeping resource usage minimal). The price determination process (Figure 9.2, step 4) is dependent on the length of the estimated job

execution time, amount and type of resources available during that interval, time of day, and the resource pricing profile. Upon completion of bid creation the tender broker will place the bid with the requesting broker (Figure 9.2, step 6). Note that this round of the bidding process is a closed bid whereby all tender brokers have no prior knowledge of the budget allocated for this job.

At the requesting broker only bids received prior to the bid closing time will be considered during the bid evaluation phase (Figure 9.1, step 5). Once the bid closes, all submitted bids will be analyzed based on the optimization settings of that particular job (Figure 9.1, step 8). In cases where there are submitted bids meeting all the job requirement constraints, a winning bidder may be determined based on how closely it matches the job optimization constraints. However, in cases where none of the bids matches the constraints, a re-negotiation phase may be initiated (Figure 9.1, step 9). During the renegotiation phase, only those tender brokers who have submitted bids previously (signifying that they have the available resources for the particular job) can participate. This second round of bidding is of an open bid nature where the budget allocated for that particular job will be announced. In addition, the requesting broker will be willing to negotiate as long as the new price charged by the tender broker falls within 110% (or other user specified range) of the budget originally set aside for that particular job, while the promised job completion time falls within the sum of the original deadline and a predetermined-time grace period. The requesting broker re-evaluates the new bids and selects the best match among those that meet the constraints. If no such bid is found then the job fails.

Figure 9.3 shows the proposed bidding process (the re-negotiation process is not shown as it is similar to the bidding process). The tender brokers will always attempt to match the constraints set forth by the job in an open bid, while still being able to profit from this transaction. However, if the open bid process fails then we know that no tender brokers will be

able to meet the job constraints (while still profiting from it), hence the bidding transaction fails, leading to the failure of the job.

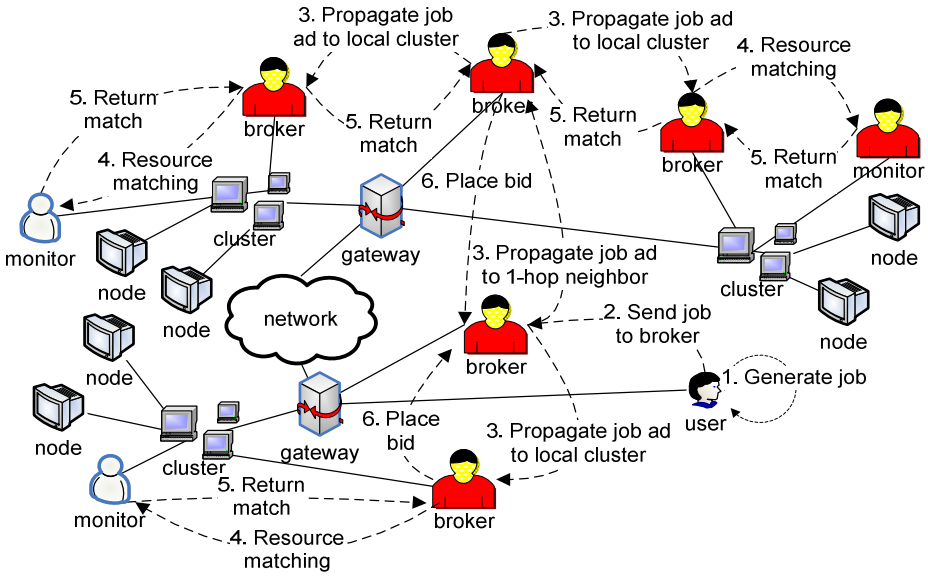

Figure 9.3 Bidding Process.

After determining the winning bidder, a contract will be sent to the winning bidder as an agreement between the brokers. Only upon signing the contract can the tender broker accept the job and begin processing the job. If for any reason the tender broker rejects the contract, the bid will be voided. A penalty price will also be included in the contract should the winning broker fail to complete the job execution by the agreed upon deadline.

9.2 Auction Based Grid Resource Scheduling Using Combinatorial Exchange Methodology

This section discusses the implementation of a combinatorial exchange for Grid resource allocation and scheduling.

9.2.1. Logical Flow of Auction Participants

At the beginning of each auction, the auctioneer first determines the type of auction to use and prepares the necessary environment prior to sending notifications to all participants of market opening. Figure 9.4 shows the initialization flow of an auctioneer.

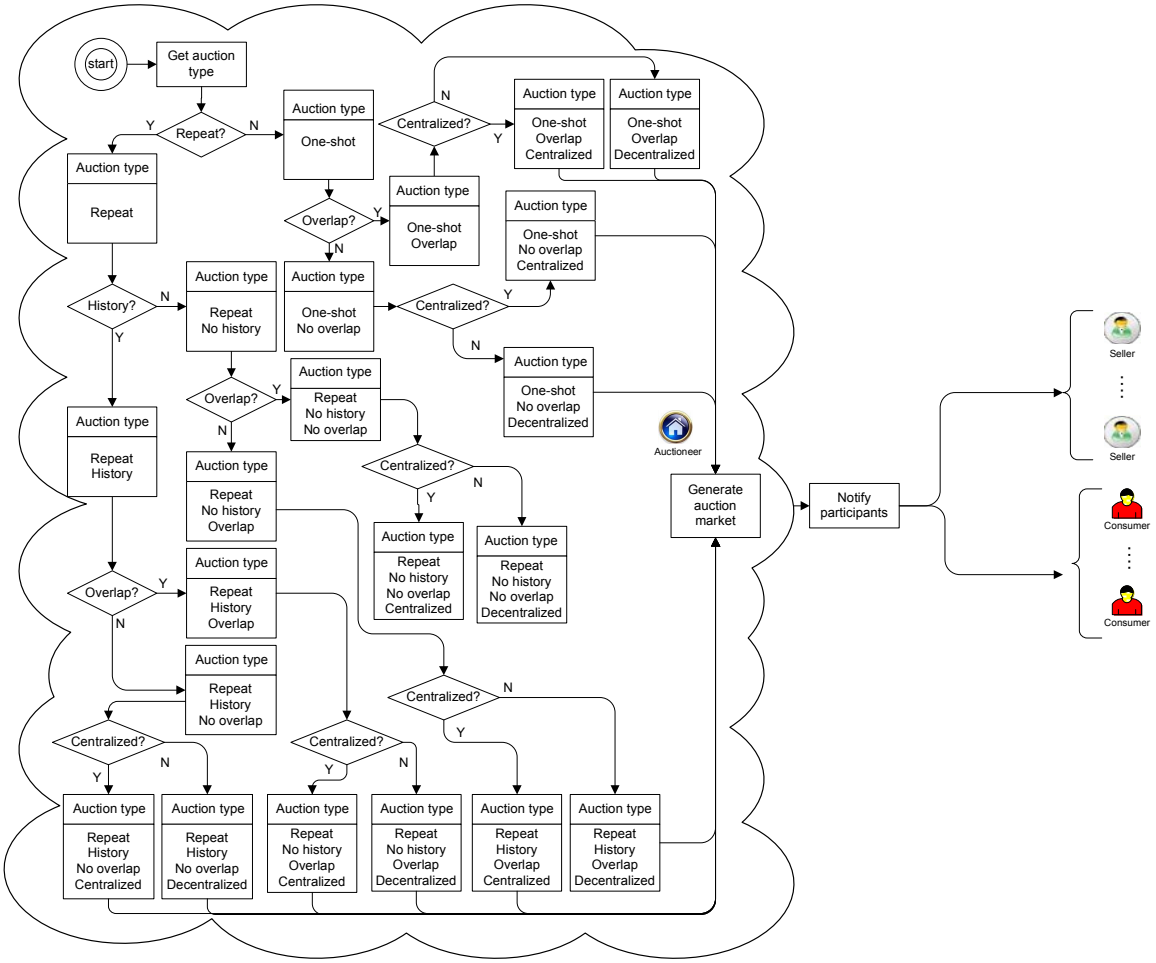

Figure 9.4 Auctioneer Initialization.

Prior to registering for auction participation with the auctioneer, each consumer performs auction market environment checking to determine the type of auction to be deployed in the market. After the necessary initialization has completed, the consumer fetches each job from storage and determines the expected execution required per job as well as the corresponding bidding strategy to adopt for that job. Once the strategy has been determined, an auction request bundle is created for each job and sent to the auctioneer. Figure 9.5 shows the logical flow of a consumer.

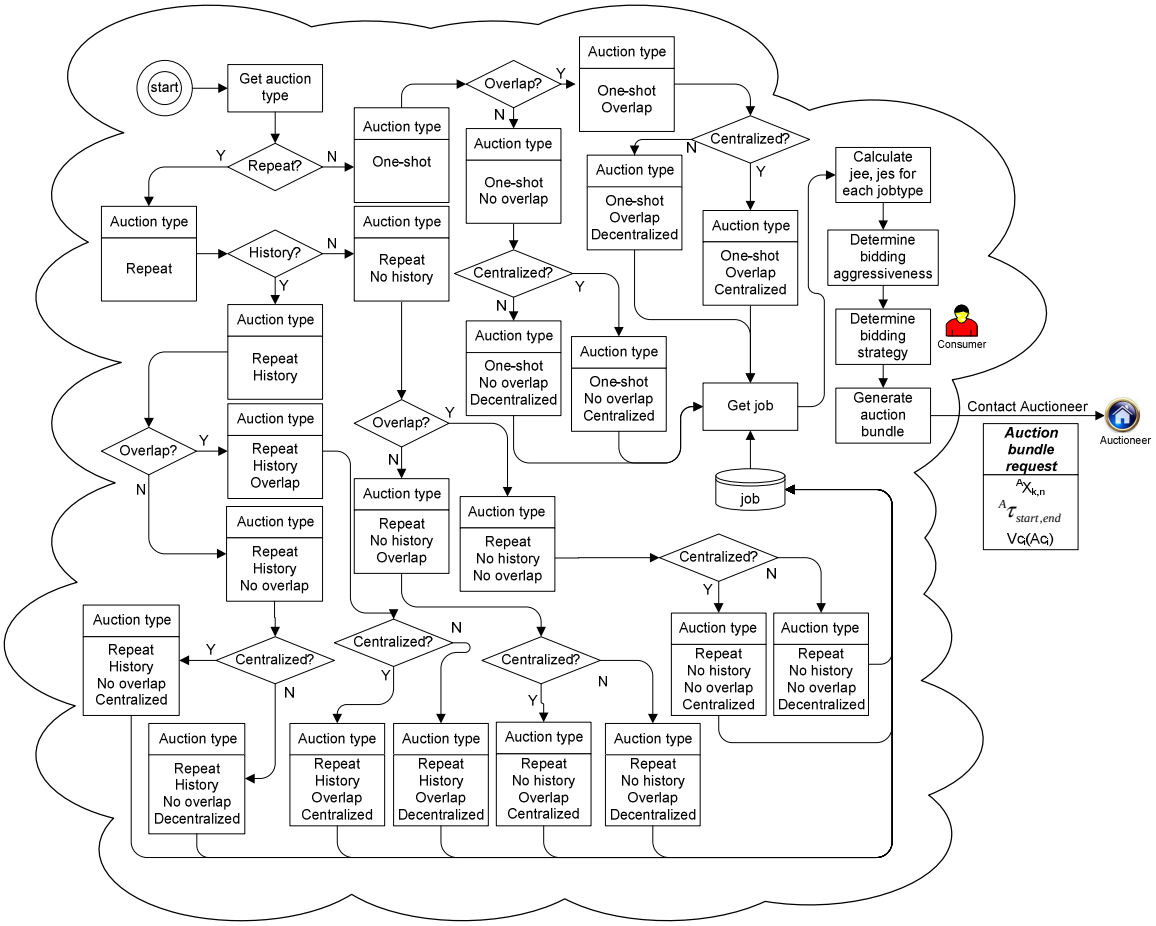

Figure 9.5 Consumer Logic.

Similar to the consumers, participating sellers first determine the type of auction to be deployed in the market, followed by collecting current information of all her resource information and status before registering with the auctioneer for auction participation. Figure 9.6 shows the initialization flow of a seller. In this phase, sellers do not make any attempts to determine bidding strategies as no information on auction bundle requests is available yet.

Upon receiving registration information from both consumers and sellers, the auctioneer determines the market price (ρ) for each resource based on all resource demands from the consumers and the overall availability of resources offered by all sellers. Once the market price for resources has been determined, corresponding consumers' initial auction bundle requests will be adjusted using the current market value. All auction participants are informed of the

current market value for all resources. The market price determination process is only initiated once for the very first auction. Subsequent market prices will be adjusted from this initial market price based on overall demand and availability of resources at any point in time. This allows fluctuations in the market due to changes in demand and supply of resources.

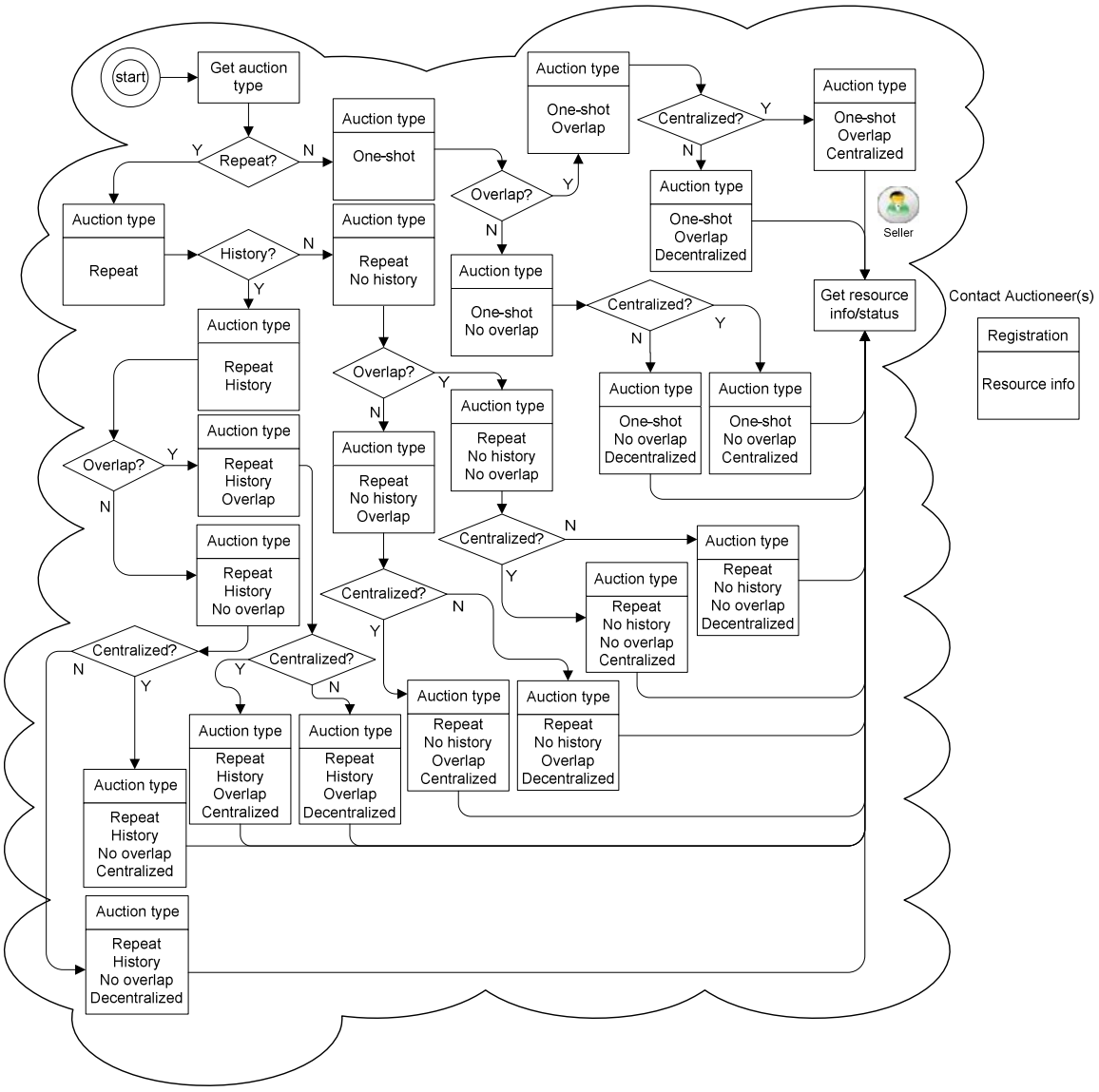

Figure 9.6 Seller Initialization.

After determining the market prices for all resources, the auctioneer propagates each consumer's auction request bundle (minus consumer's valuation for that bundle which is kept private) only to sellers who possess the required resources (based on information sent to the auctioneer when each seller first registered with the auctioneer). For the simulation, we make the assumption that sellers do not purchase new resources during the course of the simulation. But in the event that a seller adds new resources in her inventory, all she has to do is to resubmit the updated resource information to the auctioneer. At this point it is important to note that although we make the assumption that sellers do not purchase new resources, current resources may fail from time to time during the course of the simulation. This is implemented so as to simulate machines failures in real life. Figure 9.7 shows the bid handling flow of an auctioneer.

Upon receiving auction bundle requests from the auctioneer, each seller checks her current resource status to determine if she can meet the specifications described in each request. If so, the seller will determine the aggressiveness index as well as any corresponding strategies to adopt before submitting counter bids to the auctioneer. However, if the seller does not have any available resources to satisfy the auction bundle request, she will respond to the auctioneer that she will not participate in the bidding for this auction bundle. Figure 9.8 shows the auction participation flow of a seller. Sellers may resubmit bids at any point in time so long as the auction remains active. However, every bid placed incurs a participation fee c_p to deter abusing of this privilege by any auction participant.

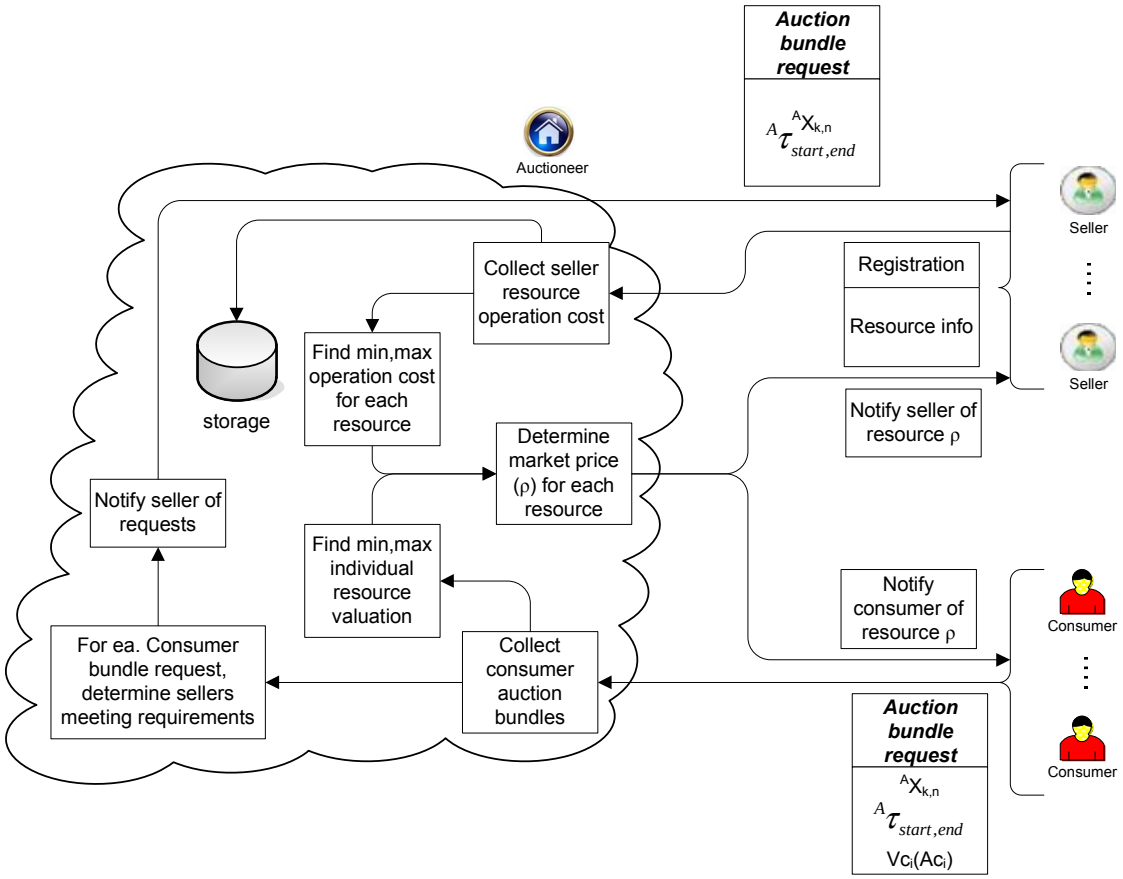

Figure 9.7 Bid Handling by Auctioneer.

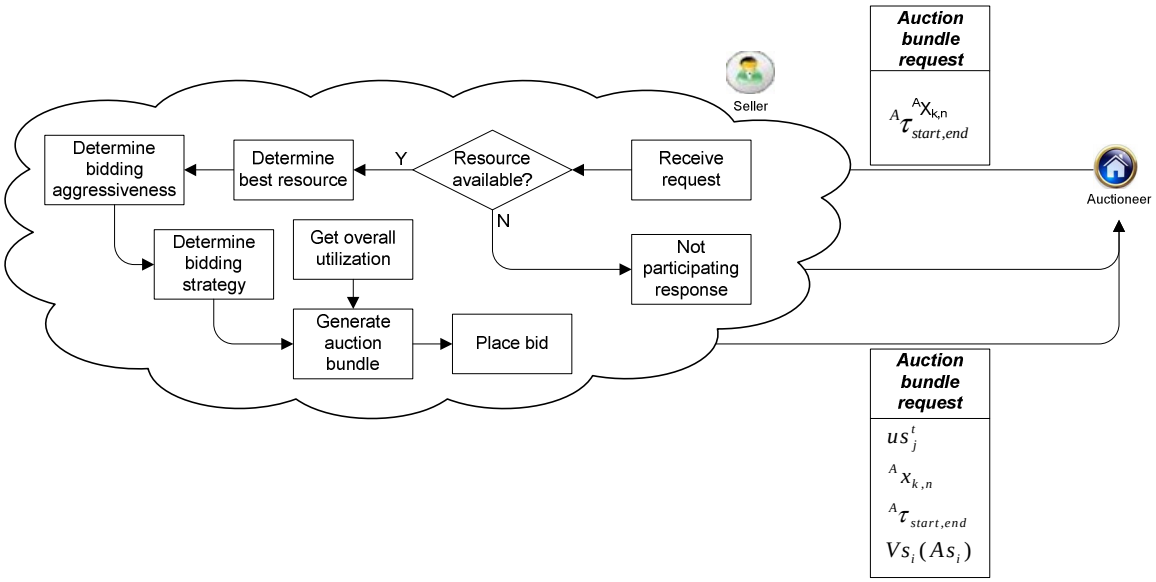

Figure 9.8 Seller Auction Participation.

At the end of every auction, the auctioneer matches consumers' auction bundle requests to sellers' offers based on the matching strategy used. There are three available strategies to the auctioneer:

I. Sellers' profit maximization,

II. Consumers' cost minimization,

III. Matching maximization – where the auctioneer attempts to satisfy a maximum number of possible matches.

Figure 9.9 shows the resource assignment flow of an auctioneer.

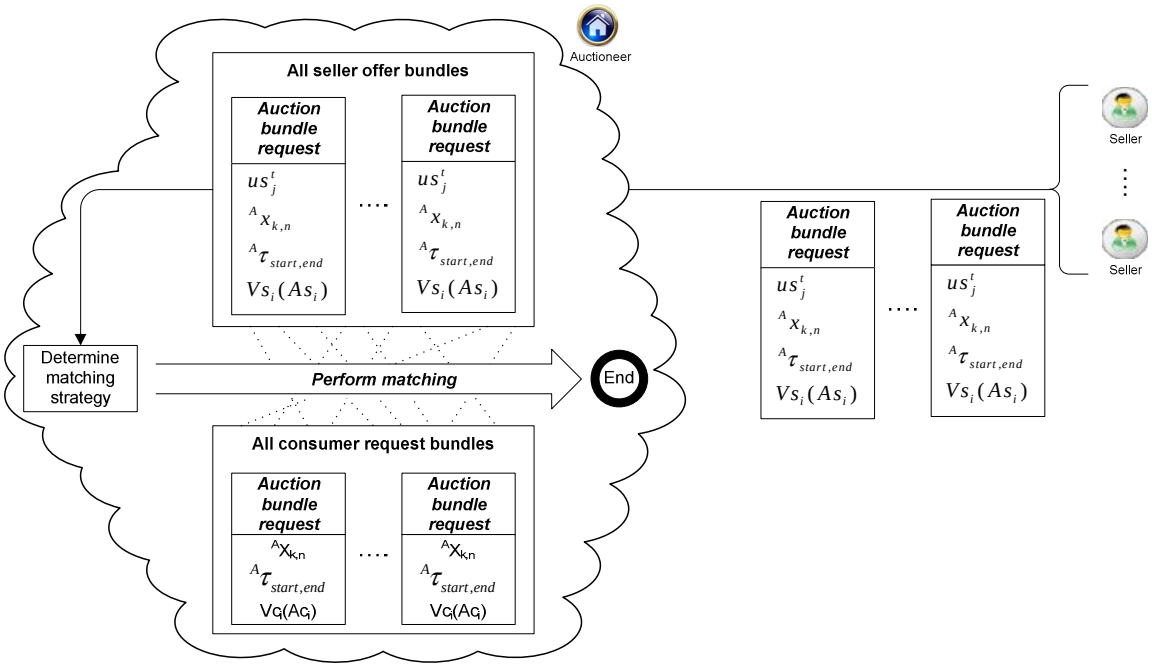

Figure 9.9 Auctioneer Resource Assignment.

9.2.2. Bidding Strategies

The bidding strategies adopted in the Repeated Auctions depend highly on past experiences. Factors such as price determination, bidding behavior, bidding aggressiveness, and which auctions to participate in (for decentralized auctions) are decided based on this information.

Prior to the start of the first round of auction, each participant (in a Repeated Decentralized scenario) is pre-assigned to a specified set of decentralized auction locations. For each subsequent round, auction participants decide where to bid and how to overlap bids in overlapping scenarios based on the collected information about each auction play. She makes the decision of which auction to participate in (for decentralized cases) by identifying where the highest utility returning auctions are. For example, since a consumer's ultimate goal is to have her jobs processed, she would prefer to participate in auctions where sellers offering the required resources reside. On top of that, sellers from past auctions who yielded the highest job utility returns (based on seller's resource offering prices) are often preferred as this usually signifies that those sellers have the cheapest available resources. Additionally, the consumers would choose to avoid auctions where the strongest competitors (for open bidding cases) reside. This is an attempt to avoid direct conflict with strong competitors, unnecessarily pushing up resource prices. However, if both best seller and worst competitor reside in the same auction, the preferred seller takes precedence as the consumer's ultimate objective is to have her jobs executed. The same applies to seller behavior. In the approach taken here, the decision for an auction location is made based on two ranked lists, arranged in decreasing order, the competitiveness and value of the auction locations with respect to the job or resource type, respectively. For example, consider a participating consumer who has the following ranked competitor and seller list of potential auction places as seen in Figure 9.10 (both lists sorted in descending order). If this consumer is currently assigned to participate in at most 3 overlapping auctions, she will choose to participate in the auctions at location '1', '5' (since the seller ranking at '5' is higher than that the competitor ranking of location '5') and '7' (location '2' is skipped since competitor at ranking location '2' ranking is higher than the seller ranking.

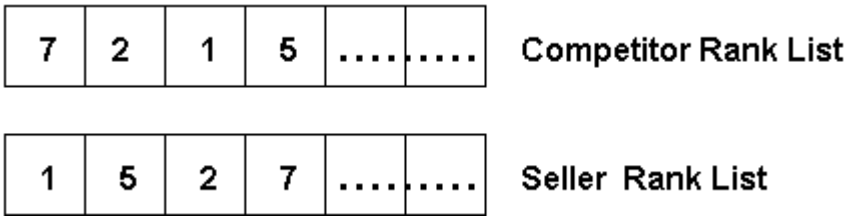

Figure 9.10 Consumer's Decentralized Overlapping Example 1.

Figure 9.11 shows another list for a different consumer. Notice that location '1' has the highest ranking in both lists. But since the seller takes precedence, this consumer will choose to participate in auctions at location '1' and '5'. Location '2' and '7' are skipped for the same reason described above. Sellers make decisions analogously.

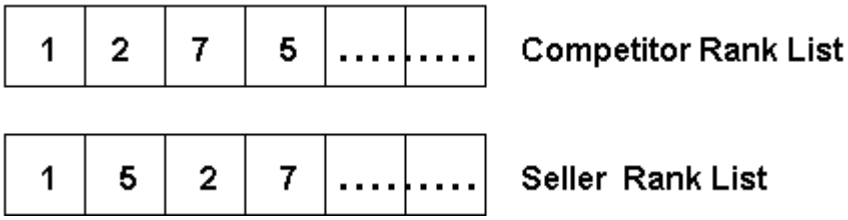

Figure 9.11 Consumer's Decentralized Overlapping Example 2.

In a non-overlapping auction, the consumers and sellers participate only in one auction place and thus simply select the location at the top of the seller list. The bidding strategies are executed by adjusting the bidding intervals and bid price increment (for open bidding). The bidding ($Aggc_i$ for consumers, $Aggs_j$ for sellers) aggressiveness is calculated based on job weight and closeness to execution deadline. The calculated value is then compared against historical bidding aggressiveness values to determine the next strategy to adopt. For example, a consumer with historical $Aggc_i$ values of 0.8, 0.92, 0.99, 1.0, 1.05, and a current $Aggc_i$ value of 1.01 would insert the new data in the 5th position in the $Aggc_i$ historical list. Since its

placement is now 2 positions from the middle value (0.99) and in the 2nd position out of the remaining 3 higher values, the bidding interval will be incremented by 2 units of the bidding step size. So if the initial counterbidding strategy is to place a bid for every 3 counterbids, it will now place a bid for every 5 counterbids as long as there is enough time before the auction ends. However, if there are no bids submitted and the auction is ending soon, the consumer will not hesitate to place a bid as long as she is not the last bidder and the bid increment is still within her reserve valuation.

The bidding price determination is based on past market value trends. For example, if a consumer witnessed that the price of resources for job type *i* in the past 3 auctions has been increasing twice and the decreased in the most recent auction, expressed as $(\uparrow,\uparrow,\downarrow)$, she would expect the next market value to be somewhat close to the closing price of the last market price since the most recent price trend $(\downarrow)$ carries more weight (a weight of 2) than the older information $(\uparrow,\uparrow)$, each carrying a weight of 1, which eventually smoothes out the fluctuations. On the other hand, if the previous information is $(\downarrow,\uparrow,\downarrow)$, resulting in a weighted trend of -1+1-2 = -2 out of 4, she would expect the market price to drop and would start the auction by bidding 50% of the originally anticipated value (i.e. the resource closing price from the previous auction). The minimum and maximum price variation is limited to 50% for downward trend, and 150% for upward trend.

CHAPTER 10

SIMULATION DESIGN

This section discusses the simulation approach for the auction mechanism in our system.

10.1 Broker and Job Advertisement Based Grid Scheduling

To evaluate our bidding based grid job scheduling we have designed and implemented a discrete event grid simulation tool in C++ (Figure 10.1 shows the architectural layout). The simulator is designed to follow a hierarchical approach where a network of routers is created followed by subsequent gateways, clusters, and nodes. Each gateway consists of a gateway broker (serving as a representative broker for the entire gateway), a monitor (for monitoring all resources within the gateway), and users (generating job requests). At the clusters, brokers are used to perform inter-cluster brokering; monitors are used to monitor resources within the cluster (and reporting to the gateway); and nodes represent individual computers (with the available resources to the cluster). In order to simulate a more realistic grid environment, various random types of cluster and node instantiation schemes have been adopted to generate sites. Within each site, overall resource status is monitored by the gateway monitor while the cluster monitor keeps track of all local nodes' types and schedules and currently used resources within the cluster. To be able to create a scenario and study the behavior of the simulation under full load, the user object is configured to constantly provide a steady stream of new jobs which keeps the broker busy. Jobs are generated using a uniform random generator with an execution time ranging from 1 hour to 5 days (based on a 2.4GHz processor). The arrival rate of jobs follows that of a Poisson arrival process.

During the bidding process, job advertisements will be submitted to all neighbors residing one hop away from the requesting host. Note, that the number of job advertisements

propagated is directly related to the number of neighbors. This is done deliberately to limit the flooding of the network by performing unnecessary broadcasting of job advertisements. To better simulate the effects of network delays in our simulation, we have included the effects of processing (assumed constant), queuing, transmission, and propagation delays. As a result, the time taken and cost of data propagation depends on the queue length, data size and distance propagated.

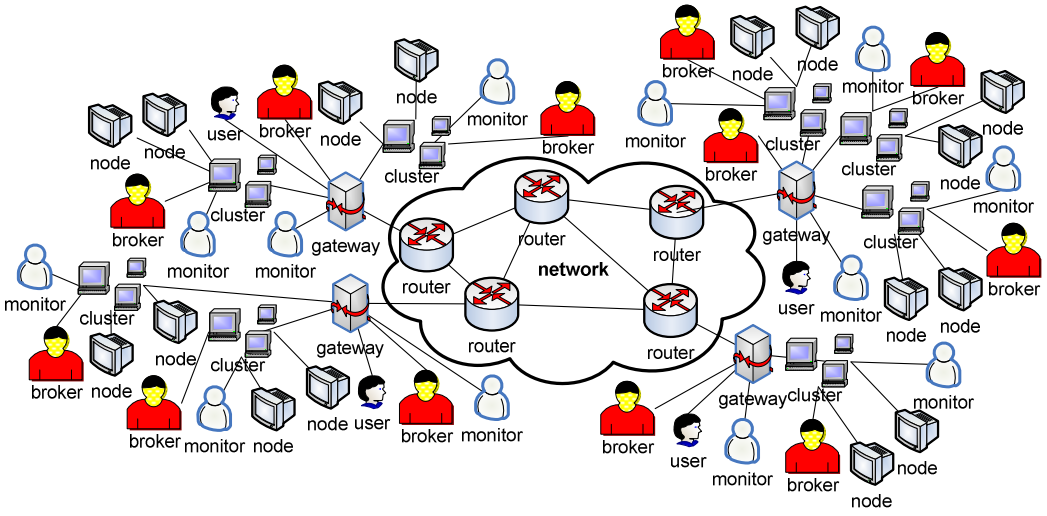

Figure 10.1 Simulator Overview.

10.1.1. Resource Management

In our computational grid model there are two main entities: requesting brokers and tender brokers. The requesting broker collects jobs to be processed from users and is willing to reward a cluster that is able to successfully complete the job within its timeframe and/or budget. The tender broker offers its service and resources to anyone who has jobs to be executed if the price is right. These two entities control the supply and demand of the grid resource market. The requesting broker is always in search for the least expense way to process its jobs while getting results at the earliest-time. The tender broker is always trying to maximize its earnings while minimizing usage of its own resources so as to have more available resources to process other jobs. As a result, efficient resource management becomes self motivational as each participant

is always trying to get the best “bang for the buck” while being thrifty in its own resource usage. It is important to note that in a typical trading environment there is always a maximum price a requesting host is willing to pay for a particular service. And similarly, there is always a minimum price which the service provider is willing to accept for offering its services. Although renegotiations may be performed in an attempt to arrive at an acceptable price between the requesting and tender brokers, it is possible that sometimes such renegotiations may fail as such an acceptable price may not be attainable. In such cases, the trading process fails.

10.1.2. Price Formulation

In order to mimic the variations in the computational grid system load throughout the day, a simplified model has been adopted to be used as a reference to predict the usage of resources at different times of the day. Figure 10.2 depicts the simplified model used to predict system load throughout the day over a 2 day period. This model represents an estimated system load throughout a day; thus we will be able to estimate the total load between two given times of the day by performing an integration function from the start time to the end time. This integration result is then multiplied with the respective costs of all the resources required by that particular job over the period of estimated execution time. After obtaining the initial estimated cost, this information is used in conjunction with the reserved resources (within that cluster) for all jobs scheduled to be executed during the same period to estimate the load on that particular cluster. The estimate of the overall resources used at that cluster during the execution period is then divided by the total resources available at the cluster, which is compared to the same fraction when the cluster is 50% utilized. The resulting difference is used to scale the initial estimated cost of processing that job. For example, if the initial estimated price is 2000 units, and the estimated overall system utilization during the job execution period is calculated to be 70%, then the price would be scaled up by 20% to 2400 price units.

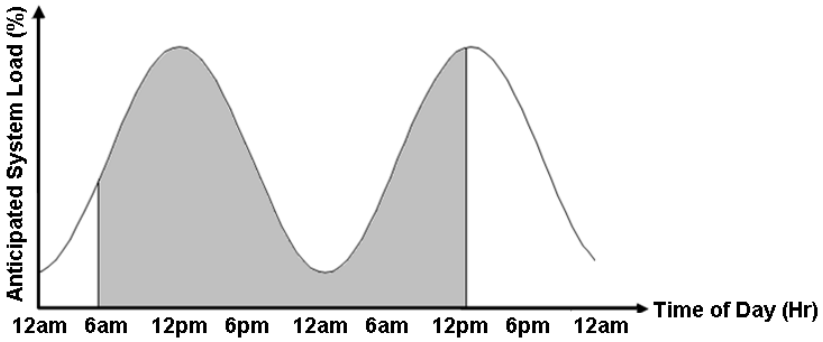

Figure 10.2 Grid Anticipated System Usage Model.

In order to evaluate the effectiveness of this scheme, two other schemes have also been implemented as bases for comparison:

The first is a conventional grid scheduling scheme (denoted as conventional from here on) similar to that of the ATLAS project. Instead of manual site selection by the administrators, this scheme performs random site selection for job execution. This helps in distributing the load of job executions across the system. Note, that clusters and nodes within each site in the conventional scheme are heterogeneous. By simulating clusters with varying resources, we can improve on the number of successful resource matches and hence serve as a more competitive comparison with our proposed bidding scheme.

The second scheme (denoted as random from here on), is similar to the conventional scheme. However, this scheme has the ability to perform renegotiations to find a better resource match within a particular site.

CHAPTER 11

SIMULATION RESULTS AND SUMMARY

This section discusses the simulation results from the auction mechanism in our system, and summarizes the findings based on those results.

11.1 Results From Broker And Job Advertisement Based Grid Scheduling

Simulated data from the three previously described schemes were collected and compared. Data collected from the conventional and the random schemes are used for comparison to our proposed bidding scheme. Throughout its lifetime, a job may be in one of several stages in its execution phase: queued, staging-in (transferring the required input data sets to the executing host), executing, staging-out (transferring the resulting data sets to the designated receiving host), and successfully executed. A job is considered to have successfully executed only upon completion of the actual job execution process, with the staging-out of resulting data files delivered. In order to simplify the job stage categorizing process, we have separated them into 2 categories: successfully executed and successfully matched. Successfully matched implies that the necessary matching and bidding processes have been completed and the job may be in any one of the previously mentioned stages other than successfully executed. However, a job match may fail and be marked as a match-failed. Match-failed reasons include: i) failure to match the job resource requirements to an available host; ii) failure to meet the budget requirements set forth by the requesting broker; iii) failure to meet the execution deadline requirements; iv) failure to place bids prior to auction closing; or v) failure to sign and return the contract issued by the requesting broker.

Figure 11.1 presents the total number of jobs executed over a period of 30 days. As more resources get tied up with job execution during the course of the simulation, a decreasing number of resources remain available. As a result, it becomes increasingly difficult to find the

required resources at the same low price as before. As observed from Figure 11.1, the bidding scheme outperformed the other two schemes by delivering the highest number of successfully executed jobs over the simulated period. The main reason for this phenomenon can be attributed to the fact that bidding possesses the advantage of being able to obtain bids from all one-hop neighbors, hence increasing the chances of a job match. The conventional scheme performs worse than the random scheme due to lack of renegotiation ability. For a better insight, we refer to Figure 11.2 which shows the total number of successful renegotiations achieved for the random and the bidding schemes. When a job match fails, renegotiation is performed between the requesting broker and tender broker. Default settings for the renegotiation process in our simulation limits the variation from the original time and budget to 10%. As observed, the random scheme fails to take full advantage of the renegotiation tools due to the limited availability of resources within the single randomly selected site. If the chosen site is already overwhelmed with jobs, further renegotiations may not necessarily improve the matching of jobs to resources. At this point, it is important to note that although the bidding scheme achieved approximately twice as many successful renegotiations as that of the random scheme, the number of jobs attaining the successfully executed state (see Figure 11.1) by the bidding scheme did not perform twice as well when compared to the random scheme. This is because although more jobs are matched with renegotiated resources, eventual successful execution of jobs still depends on additional factors such as network traffic, data and output file staging processes, and resource availability at the moment in time when jobs are to be executed.

Figure 11.3 shows the accumulated monetary transactions collected throughout the simulation. The bidding scheme performs dramatically better than both the random and the conventional schemes. This can be attributed to the profit maximizing characteristics of the bidding scheme, along with substantial improvement in the number of successful job matches due to re-negotiations (see Figure 11.2). Although the random scheme also supported

renegotiation, it is limited by the number of available resources from its single renegotiating location.

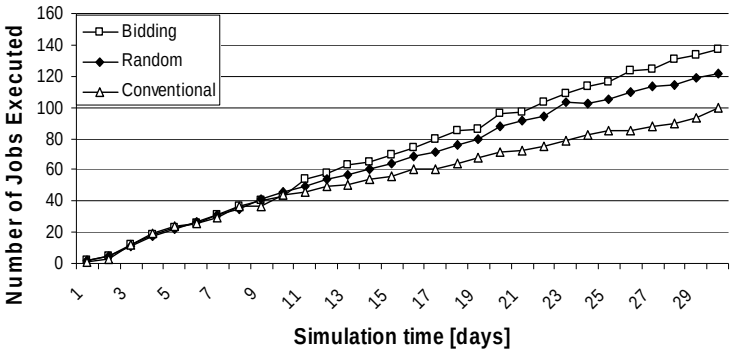

Figure 11.1 Total Number of Jobs Executed.

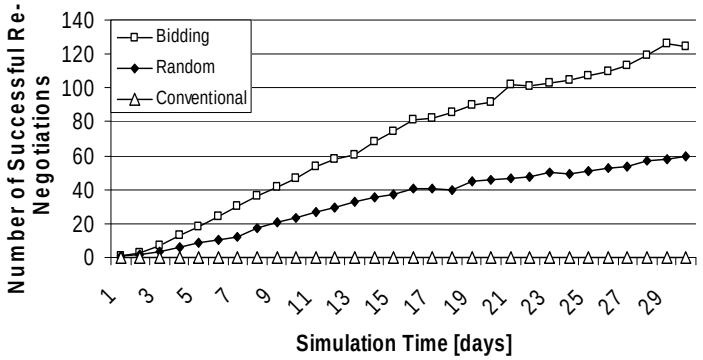

Figure 11.2 Total Number of Successful Re-negotiations.

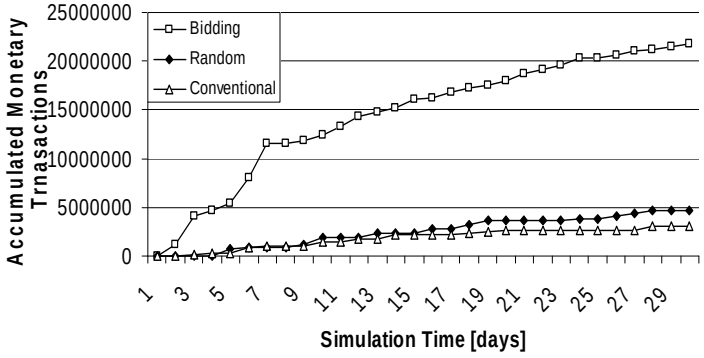

Figure 11.3 Accumulated Monetary Transactions.

Figure 11.4 shows the total penalty resulting from exceeding the promised estimated job completion time, as stated in the binding contract. The current penalty implementation in our simulations is simple: if a site fails to abide by the contract, 30% of the initial contract price is returned to the requesting broker. Although the bidding scheme resulted in the highest accumulated penalty, if we look at the penalty to earning ratio graph of Figure 11.5, we can see that it remained within 14% of the overall earnings throughout the simulation process. The conventional scheme, due to its limited number of successfully matched jobs (and hence limited penalties), performed better in terms of paying for penalty charges, when compared to the random scheme; it managed to limit the penalties paid to within 18% of its accumulated earnings. Although the random scheme paid fewer penalties as compared to the bidding scheme, it was the worst performer of the group by paying up to 26% of its total earnings to penalty charges.

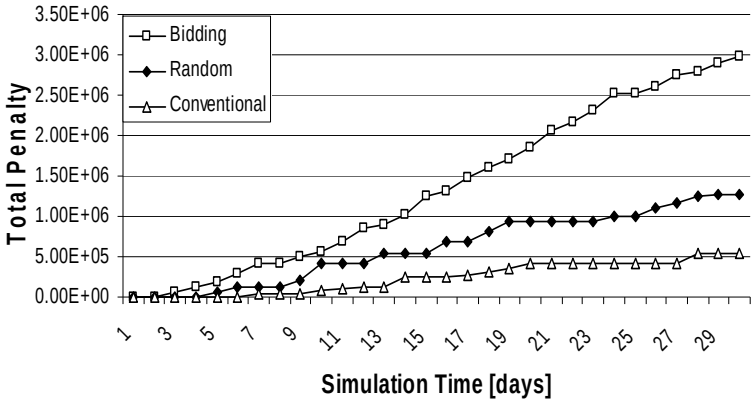

Figure 11.4 Total Penalty.

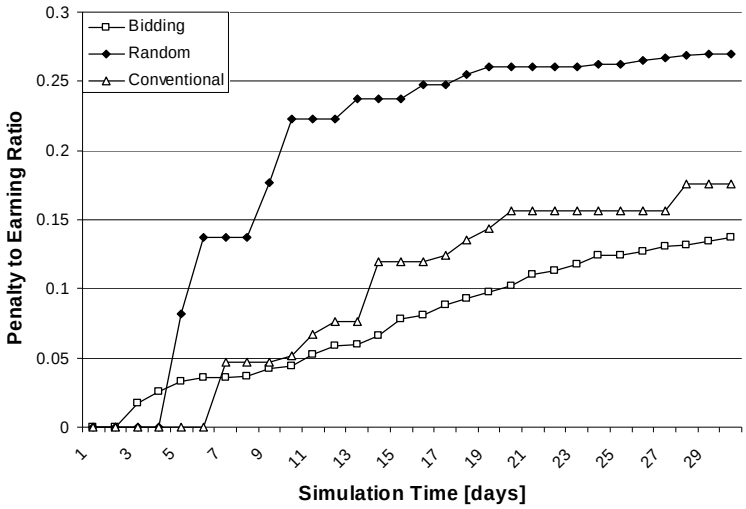

Figure 11.5 Penalty to Earning Ratio.

11.2 Results from Auction Based Grid Resource Scheduling Using Combinatorial Exchange

In this subsection we will discuss the simulation findings of the different types of auctioning schemes. As the One-shot auctioning scheme is simply the first case of a repeated auctioning mechanism, we will focus on analyzing the Repeated auctioning schemes here.

Figure 11.6 shows the comparison between successful and failed attempts at assigning consumer jobs to seller resources for different auctioning mechanisms. The simulation was performed with each consumer generating a job request approximately every 0.2 units of time to ensure that the system is overwhelmed with job requests throughout the simulation process. From Figure 11.6, we can see that every auctioning scheme has been heavily overloaded with job requests. Figure 11.7 shows the overall resource utilization of all sellers' resources. Participants in a Centralized auctioning scheme have the advantage of achieving higher resource utilization simply because they have access to all jobs and resources within the auction. In a decentralized scheme, there is always a possibility that a generated job requires a specific combination of resources not available within that particular decentralized auctioning group. If there are too many failed auctions in an overlapping auction group, participants in such a group may opt to join other overlapping auctions in subsequent auctioning rounds in an

attempt to avoid further occurrences of non-matching job requests. Similarly, participants who continuously lose out in an auction may also choose to join another overlapping auctioning group. For example, if a seller is constantly being outbid by another seller offering lower prices for her resources, the losing seller may intentionally leave this auctioning group in order to avoid future competition with this competing seller. This is reflected in the utilization data in Figure 11.7 by the fact that overlapping distributed auctions achieve higher resource utilization compared to no overlap distributed auctions.

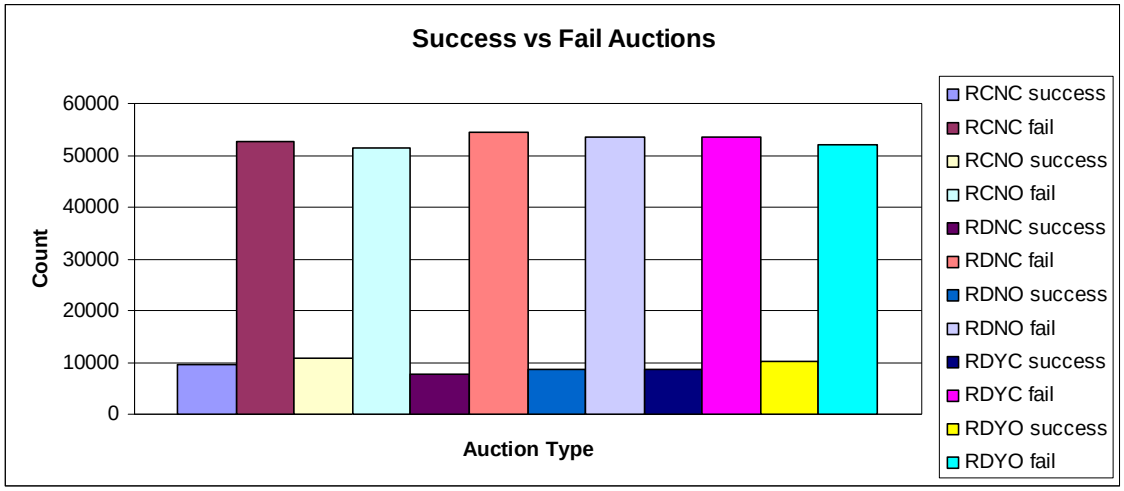

Figure 11.6 Success vs. Failure Comparison

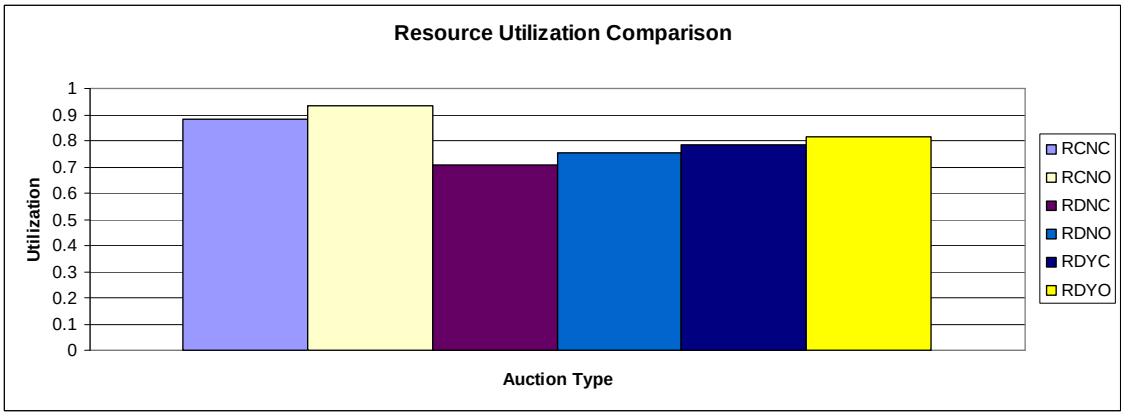

Figure 11.7 Overall Resource Utilization.

From this data, it can also be observed that open auctions tend to lead to better resource utilization than the corresponding closed bid auction scheme.

Figure 11.8 compares the total number of bids submitted for different auctioning mechanisms. Since there is a participating cost associated with every bid submitted, it would always be good to win an auction with the minimum number of bids submitted. Furthermore, the number of bids submitted not only adds to the participation cost, it also leaves a smaller profit in return. Figure 11.8 shows that the centralized auctioning scheme places the highest number of bids when compared to decentralized auctioning schemes. This is mainly because a centralized auctioning environment faces more job requests (and resource offers) as compared to that of a decentralized environment. In addition to that, they will face greater competitions in such an environment as well.

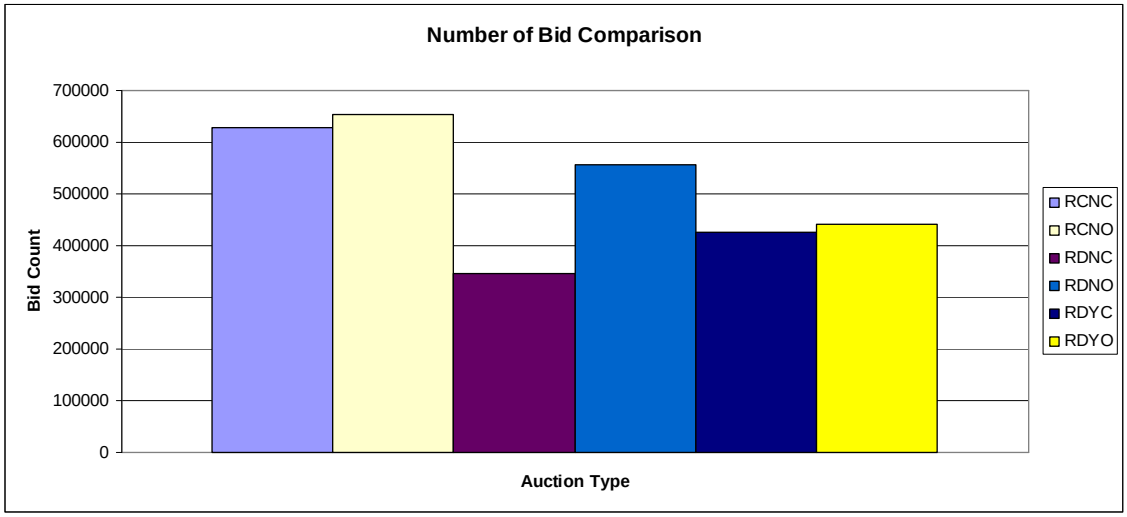

Figure 11.8 Auction Bids Comparison.

Besides influencing costs, the number of bids also represents a measure of the communication overhead imposed by the different auction-based scheduling schemes. As the number of bids at a given auctioneer increases, so does its computational load and bandwidth requirement, eventually slowing down the scheduling and introducing a bottleneck. The results presented in Figure 11.8 show that the distributed auction mechanism results in a significantly

lower number of total bids, thus reducing the amount of communication required for scheduling. Moreover, since these bids are distributed across 10 auctions, the bandwidth and computation overheads at the scheduler are actually reduced to less than $\frac{1}{10}$ of the ones for the centralized auction-based schedulers, demonstrating the scalability of the distributed auction approach. Figure 11.9 depicts the winning to reserve price comparison. This is useful when comparing the value returns from winning jobs under different auctioning mechanisms. In this diagram, we can see that the Repeated Centralized Open CE (RCOCE) mechanism yields the lowest returns. This is mainly due to the fact that it faces the most competition from all participants within the auctioning environment. Among all decentralized auctioning mechanisms, the Repeated Decentralized Open CE (RDOCE) without overlap scheme performs the worst. This may be attributed to the reason that it faces the most competition with other participants without the ability of switching to other auctioning groups. Although the Repeated Decentralized Closed CE (RDCCE) without overlapping scheme retains a better ratio in this figure, we can see from Figure 11.10 that it also has the lowest success ratio in an auction. In other words, it has the lowest average number of jobs being scheduled throughout the auction.

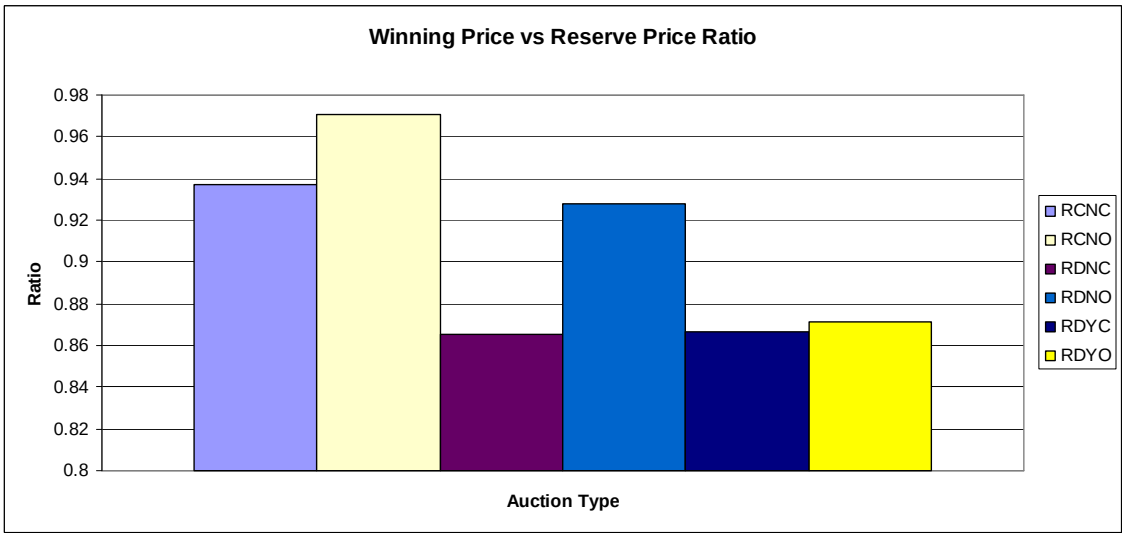

Figure 11.9 Winning Price vs Reserve Price Comparison.

From Figure 11.10 which illustrates the auction success rates, it can be seen that the RCCE without overlapping enjoys the highest auction success ratio. But Figure 11.11 shows that it has one of the lowest value returns per job. This is mainly because of the centralized bidding nature when jobs are available to all potential competitors within the auctioning environment. Figure 11.10 also shows that, as would be expected, centralized auctions generally have a higher auction success rate as compared to decentralized schemes since they have access to all available resource bundles for all incoming bids, thus having the opportunity to make the most optimal job to resource matches. In contrast, jobs in a distributed auction can be matched only to resources available at the specific auction site, increasing the risk that the needed resources are not available. The data in the figure, however, also shows that the distributed open auction with overlap can partially overcome this limitation, leading here to a job completion rate that is comparable to the centralized auction schemes. This is due to the fact that consumers and sellers can successfully change auction location based on the past performance data, allowing them to form auctions with higher success ratios. Figure 11.11 shows a comparison of the valuation of the jobs scheduled in each auctioning scheme. Surprisingly, the Repeated Centralized Closed CE (RCCCE) without overlapping has the lowest value-per-job returns. This may be due to its relatively high winning price vs reserve price ratio (Figure 11.9), as well as to the high number of bid counts (Figure 11.9) which results in a significant value reduction. As such, it may be derived that the RCCCE without overlapping auctioning mechanism is paying a high price for bid participation while not winning enough auctions to cover the participation costs.

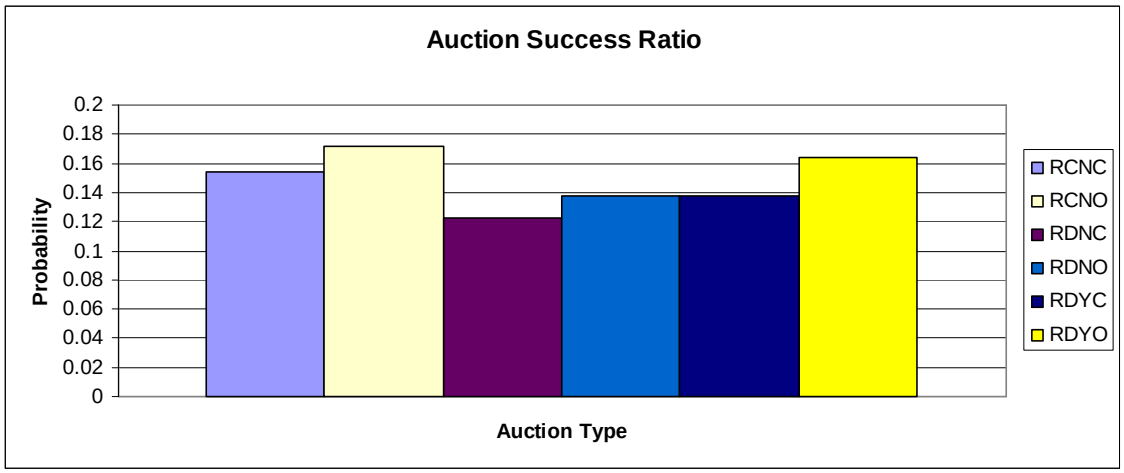

Figure 11.10 Auction Success Comparisons.

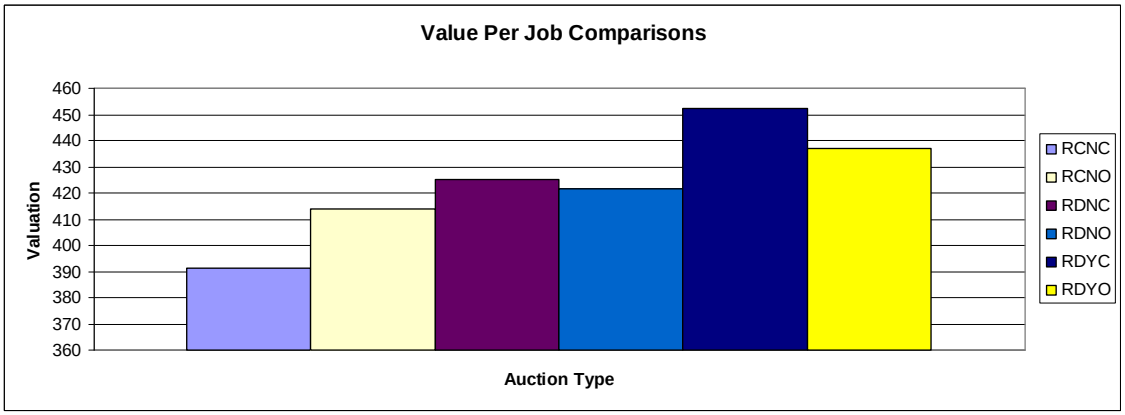

Figure 11.11 Job Value Comparisons.

It is observed that the Repeated Decentralized overlapping schemes yields the highest value per job returns. This may be due to the strategy of focusing on participating only in auctioning groups where one has a clear advantage over the other competitors. As such, each decentralized auctioning group slowly becomes a specialized auctioning group where different groups specialize in different types of auctions, yielding an increase in success rates as well as in scheduled job value. Overall, these results show that the auction-based scheduling

mechanisms are successful at scheduling jobs for grid applications. Moreover, they demonstrate that the distributed auction schemes can effectively reduce communication and scheduling overheads by reducing the number of bids needed to obtain job to resource matches. While this initially leads to decrease in success rates, the data also shows that the distributed open bid auction scheme with overlap can address this by allowing consumers and sellers to establish custom auction places, leading eventually to success rate comparable to the centralized auction schemes. In addition, the results show that the decentralized schemes, due in part to the reduced need for bids and more focused auctions outperform the centralized approaches in terms of job value scheduled and winning to reserve price ratio. Together this demonstrates the potential of the decentralized combinatorial exchange model to address the issues in grid job scheduling without creating a scheduling “bottleneck”.

CHAPTER 12

FUTURE WORK

This section discusses findings from simulation studies, as well as additional work that may be done in the future.

12.1 Broker and Job Advertisement Based Grid Scheduling

In the study of broker and job advertisement based Grid scheduling, we have used a simple, flat 30% penalty scheme. Our intentions are to implement a flexible penalty scheme with demerit scheme where winning brokers who break the contract for any reasons will have a demerit point added to the rejecting broker for either backing out of a winning bid or failing to abide by the contract. This would in turn affect their ability to participate in subsequent future biddings. We are also looking into improving the current scheme to incorporate multiple-hop job advertisement propagation ability, support for multiple auctioning and better decision-making tools, along with the ability to subcontract jobs as well as better job QoS and prioritization support.

12.2 Auction Based Grid Resource Scheduling Using Combinatorial Exchange Methodology

In the study of the different CE auctioning mechanisms, the decentralized auctioning scheme has a clear advantage of minimizing the auctioning overhead while yielding higher job value returns when compared with their centralized counter parts. The ability to overlap with other auctioning environments allows participants to move from one less desired auction to another preferred auctioning environment, hence helping in gaining bidding efficiency.

For future work, we would like to further improve upon the bidding strategies by adopting better reasoning and prediction tools, as well as by studying the effects of fluctuating load on the auctioning mechanisms.

CHAPTER 13

CONCLUSION

In this work we seek to improve resource utilization and address the common problem of scheduling and resource allocation in a computational grid system by applying Combinatorial Exchange with various auctioning mechanisms in an economic Grid environment.

Our objective was to define the technical and economical characteristics of a market-oriented Grid mechanism, to design a market-oriented Grid mechanism and to evaluate the proposed decentralized market-oriented Grid auctioning mechanism. Our contribution includes identifying and formalizing the technical characteristics of an economic grid system, proposing and designing an auctioning mechanism suitable for the grid system, and evaluating of how such an auctioning mechanism would perform in a grid environment. We evaluated and analyzed the behavioral and performance aspects of various Combinatorial Exchange (CE) auction mechanisms, including: Centralized open and closed auctions, Decentralized open and closed auctions. We also looked at the effects of an overlapping vs non overlapping auction scenario for the case of distributed auctions in a grid environment. In a set of simulation experiments we have shown the potential of a decentralized combinatorial exchange in efficiently addressing performance and resource allocation issues in Grid scheduling while avoiding the creation of a “bottleneck” as in most centralized scheduling mechanisms. These results clearly demonstrate that auction theory is a viable alternative to the traditional grid scheduling and resource allocation methodologies.

As the application of auction theory in the area of grid studies is still fairly new, much work still has to be done to further improve upon what we have done so far. In particular, further studies should be done to evaluate the specific behavior characteristics and scaling properties of the Combinatorial Exchange mechanism in the context of different loads, job and resource

types, and grid applications. Also, different bidding strategies and proxy bidding schemes could be tested in order to optimize the overall system performance.

We have demonstrated that auction mechanism, when applied to grid scheduling and resource allocation is viable, effective, and provides good performance compared to other schemes. Auction theory is relatively new in economics and as auction theory develops, new methods may provide even more benefits when applied to grid scheduling. We show that the formal model described accurately predicts actual results from scheduling very large PanDA work flows. We hope that future grid scheduling in large workflows, such as PanDA, will benefit from application of auction mechanisms.

REFERENCES

1. D. Abramson, R. Buyya, J. Giddy, “A computational economy for Grid computing and its implementation in the Nimrod-G resource broker”, Future Gener. Comput. Syst. 18(8), 1061-1074 (2002)
2. A. AuYoung, B. Chun, A. Snoeren, A. Vahdat, “Resource Allocation in Federated Distributed Computing Infrastructures”, Proceedings of the 1st Workshop on Operating System and Architectural Support for the Ondemand IT InfraStructure, Oct 2004
3. B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Shneidman, A. C. Snoeren, A. Vahdat, “Mirage, A Microeconomic Resource Allocation System for Sensornet Testbeds”, Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors, May 2005
4. B. A. Huberman, K. Lai, L. Fine, ”Tycoon: A Distributed Market-based Resource Allocation System”, Technical Report arXiv:cs.DC/0404013, HP Labs, Palo Alto, CA, USA, April 2004
5. K. Lai, L. Rasmusson, E. Adar, L. Zhang, B. A. Huberman, “Tycoon: An Implementation of a Distributed, Market-based Resource Allocation System”, Multiagent Grid Systems, 1(3):169–182, 2005
6. C. Li, L. Li, Z. Lu, “Utility Driven Dynamic Resource Allocation Using Competitive Markets in Computational Grid”, Advances in Engineering Software, 36(6):425–434, 2005
7. M. Wieczorek, S. Podlipnig, R. Prodan, T. Fahringer, “Applying double auctions for scheduling of workflows on the Grid”, Conference on High Performance Networking and Computing, Proceedings of the 2008 ACM/IEEE conference on Supercomputing,
8. W. H. Bell, D. G. Cameron, R. Carvajal-Schiaffino, “Evaluation of an Economy-Based File Replication Strategy for a Data Grid”, Proceedings of the 3rd International Symposium on Cluster Computing and the Grid
9. J. Sherwani, N. Ali, N. Lotia, Z. Hayat, R. Buyya, “Libra: a Computational Economy-based Job Scheduling System for Clusters”, Software Practice and Experience, 34(6):573–590, 2004
10. R. Wolski, J. S. Plank, J. Brevik, T.Bryan, ”G-commerce: Market Formulations Controlling Resource Allocation on the Computational Grid”, IPDPS ’01: Proceedings of the 15th International Parallel and Distributed Processing Symposium. IEEE, San Francisco (2001) April
11. C. S. Yeo, R.Buyya, “Pricing for Utility-driven Resource Management and Allocation in Clusters”, ADCOM ’04: Proceedings of the 12th International Conference on Advanced Computing and Communications, pp. 32–41. Allied Publishers: New Delhi, India (2004) December
12. A. AuYoung, L. Grit, J. Wiener, J. Wilkes, “Service Contracts and Aggregate Utility Functions”, HPDC ’06, Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing, pp. 119-131 June 2006

13. B. N. Chun, D. E. Culler, ”User-centric performance Analysis of Market-based Cluster Batch Schedulers”, CCGRID ’02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, p. 30. IEEE Computer Society,Washington, DC, USA (2002)
14. D. E. Irwin, L. E. Grit, J. S. Chase, “Balancing Risk and Reward in a Market-based Task Service”, HPDC ’04: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, pp. 160–169. IEEE Computer Society, Washington, DC, USA (2004)
15. F. I. Popovici, J. Wilkes, “Profitable Services in an Uncertain World”, SC ’05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, pp. 36. IEEE Computer Society, Washington, DC, USA (2005)
16. A. Opitz, H. König, S.Szamlewska, “What Does Grid Computing Cost?”, Journal of Grid Computing, Volume 6, Number 4, Pages: 385-397, ISSN: 1570-7873, Springer Verlag, Germany, Sept. 2008
17. J. Broberg, S. Venugopal, R. Buyya, “Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions”, Journal of Grid Computing, Volume 6, Number 3, Pages: 255-276, ISSN: 1570-7873, Springer Verlag, Germany, Sept. 2008
18. A. Kapteyn, “Utility and economics”, De Economist, Springer Netherlands 0013-063X (Print) 1572-9982, pp 1-20
19. J. A. Rodrigues-Neto, “Representing roommates’ preferences with symmetric utilities”, Journal of Economic Theory, Volume 135, Issue 1, July 2007, Pages 545-550
20. I. Foster, C. Kesselman, “The Grid 2 – Blueprint for a New Computing Infrastructure”, The Elsevier Series in Grid Computing, 2004
21. O. Shy, “How to Price: A Guide to Pricing Techniques and Yield Management”, Cambridge University Press; illustrated edition edition (January 14, 2008)
22. R. P. Stanley, “Enumerative Combinatorics, Vols. 1 ”, Cambridge University Press. (1997) ISBN 0-521-55309-1
23. W. D. Blizard, "Multiset theory," Notre Dame Journal of Formal Logic, Volume 30, Number 1, 1989 Winter: pp. 36-66
24. D. K. Gode, S. Sunder, “Allocative Efficiency of Markets with Zero-Intellience Traders: Market as a Partial Substitute for Individual Rationality”, Journal of Political Economy, 1993, vol. 101, no. 1
25. M. Xia, J. Stallaert, A. B. Whinston, “Solving the Combinatorial Double Auction Problem”, European Journal of Operational Research 164 (2005) 239-251
26. D. C. Parkes, “Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency” Ph.D thesis, University of Pennsylvania, Department of Computer and Information Science
27. P. Cramton, Y. Shoham, R. Steinberg, “Combinatorial Auctions”, 2006 Massachusetts Institute of Technology
28. S. Parsons, M. Marcinkiewicz, and J. Niu. Everything you wanted to know about double auctions but were afraid to (bid or) ask. Technical report, Department of Computer & Information Science, Brooklyn College, City University of New York, 2005.
29. J. H. Choi, H. Ahn, I. Han, “Utility-based Double Auction Mechanism Using Genetic Algorithms”, Expert Systems with Applications, vol. 34, Issue 1, January 2008, Pages 150-158
30. M. Xia, G. J. Koehler, A. B. Whinston, “Pricing Combinatorial Auctions”, European Journal of Operational Research, Volume 154, Issue 1, 1 April 2004, Pages 251-270
31. D. Levine, “Distributed Computational Biology: Clusters and Grids”, “Computational Genomics: Current Methods”, Pages 191-210
32. Klemperer, Paul, 1999. " Auction Theory: A Guide to the Literature," Journal of Economic Surveys, Blackwell Publishing, vol. 13(3), pages 227-86, July

33. A.T. Thor, G.V. Záruba, D. Levine, K. De, T.J. Wenaus, "VIGs: A Grid Simulation and Monitoring Tool for ATLAS Workflows," Proceedings of Many-Task Computing on Grids and SuperComputers (MTAGS), ACM/IEEE SuperComputing'08, November, 2008.
34. https://twiki.cern.ch/twiki/bin/view/Atlas/Panda
35. I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. "Toward Loosely Coupled Programming on Petascale Systems", ACM/IEEE International Conference for High Performance, Networking, Storage and Analysis (SC08), 2008
36. R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," Wiley-Interscience, New York, NY, April 1991.
37. Averill M. Law, W. David Kelton, David W. Kelton, “Simulation Modeling and Analysis”, McGraw-Hill Science Engineering
38. E. Medernach,” Job arrival analysis for a cluster in a Grid environment”, 1st Open International conference on Modeling & Simulation June 12th – 15th 2005 – ISIMA / Blaise Pascal University – France
39. C. Liu, I. Foster, “A Constraint Language Approach to Matchmaking”, presented at the 14th Int. Ws. Research
40. ATLAS project, http://www.usatlas.bnl.gov/
41. A.T. Thor, G.V. Záruba, D. Levine, "A Broker and Job Advertisement Based Grid Scheduling Framework," Proceedings of the Parallel and Distributed Computing Systems Conference (PDCS 2006), Dallas, Texas, November 13-15, 2006.
42. N. Nisan, T. Roughgarden, E. Tardos and V. Vazirani, editors, “Algorithmic Game Theory”, Cambridge University Press.
43. Condor High Throughput Computing, The Condor Project Homepage. http://www.condorproject.org/
44. R. Buyya, D. Abramson, S. Venugopal, “The Grid Economy”, Proceedings of the IEEE, vol. 93, no. 3, pp. , March 2005.
45. L. Xiao, Y. Zhu, L. M. Ni, Z. Xu, “GridIS: an Incentive-based Grid Scheduling”, 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
46. Z. Tan, “Market-based Grid Resource Allocation Using a Stable Continuous Double Auction”, Ph.D thesis, University of Manchester, School of Computer Science, 2007
47. F. Berman, H. Cassanova, M. Faerman, J. Schopf, A. Su, D. Zagorodnov, “Application Level Scheduling”, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 14, NO. 4, APRIL 2003
48. Y. C. Lee, A. Y. Zomaya, “Scheduling in Grid Environments”, Handbook of Parallel Computing: Models, Algorithms, and Applications. Chapman and Halls/CRC 2008
49. Nimrod/G http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm
50. D. Roure , M. Baker , N. Jennings , N. Shadbolt, “The Evolution of Grid”, Grid Computing: Making the Global Infrastructure a Reality
51. I. Foster, C. Kesselman, “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann, July 1998
52. Search for Extraterrestrial Intelligence (SETI@home) http://setiathome.berkeley.edu/index.php
53. Pareto efficiency http://en.wikipedia.org/wiki/Pareto_efficiency
54. C. Petrie, T. Webster, M. Cutkosky, “Using Pareto Optimality to Coordinate Distributed Agents”, AIEDAM special issue on conflict management Vol. 9, pp. 269-281, 1995
55. M. Beckmann, H. Kunzi, “Dynamic Pricing and Automated Resource Allocation for Complex Information Services”, Reinforcement Learning and Combinatorial Auctions Series: Lecture Notes in Economics and Mathematical Systems , Vol. 589, 2007, XII, 291 p., ISBN: 978-3-540-68002-4

56. S. Parsons, J. Rodríguez-Aguilar, M. Klein “Auctions and bidding: A guide for computer scientists”, ACM Computing Surveys, 2009.
57. J. Kalagnanam and D. Parkes, "Auctions, bidding, and exchange design," in Supply Chain Analysis in the eBusiness Area, Simchi-Levi, D. Wu, and Shen, Eds. Kluwer Academic Publishers, 2003.
58. http://en.wikipedia.org/wiki/Auction

BIOGRAPHICAL INFORMATION

Tengkok Aaron Thor received his Bachelor's degree in Computer Engineering from Texas A&M University in 2000, Master's and Doctoral degree in Computer Science and Engineering from University of Texas at Arlington in 2003 and 2009 respectively. Prior to his graduation, he was actively involved in the ATLAS experiment at Brookhaven National Lab (BNL) in Long Island, NY. His primary tasks at BNL include Production and Distributed Analysis (PanDA) monitoring and logging development, and PanDA server stress testing. His research interests include scheduling and resource allocation in Grid/Cloud computing, combinatorial optimization, and auction/game theory.