3-22-2013 12:00 AM

Decentralized Resource Scheduling in Grid/Cloud Computing

Ra'afat O. Abu-Rukba
The University of Western Ontario

Supervisor
Hamada Ghenniwa
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Decentralized Resource Scheduling in Grid/Cloud Computing

(Thesis format: Monograph)

by

Ra'afat Abu-Rukba

Graduate Program in Engineering Science
Department of Electrical and Computer Engineering

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

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

In the Grid/Cloud environment, applications or services and resources belong to different organizations with different objectives. Entities in the Grid/Cloud are autonomous and self-interested; however, they are willing to share their resources and services to achieve their individual and collective goals. In such open environment, the scheduling decision is a challenge given the decentralized nature of the environment. Each entity has specific requirements and objectives that need to achieve. In this thesis, we review the Grid/Cloud computing technologies, environment characteristics and structure and indicate the challenges within the resource scheduling. We capture the Grid/Cloud scheduling model based on the complete requirement of the environment. We further create a mapping between the Grid/Cloud scheduling problem and the combinatorial allocation problem and propose an adequate economic-based optimization model based on the characteristic and the structure of the Grid/Cloud. By adequacy, we mean that a comprehensive view of required properties of the Grid/Cloud is captured. We utilize the captured properties and propose a bidding language that is expressive where entities have the ability to specify any set of preferences in the Grid/Cloud computing environment. The language is to also enable entities to express structured preferences directly. We propose a winner determination model and mechanism that utilizes the proposed bidding language and finds a scheduling solution. Our proposed approach integrates concepts and principles of mechanism design and classical scheduling theory. Furthermore, we argue that in such open environment privacy concerns by nature is part of the requirement in the Grid/Cloud. Hence, any scheduling decision within the Grid/Cloud computing environment is to incorporate the feasibility of privacy protection of an entity. Each entity has specific requirements in terms of scheduling and privacy preferences. We analyze the privacy problem in the Grid/Cloud computing environment and propose an economic based model and solution architecture that provides a scheduling solution given privacy concerns in the Grid/Cloud. Finally, as a demonstration of the applicability of the approach, we apply our solution by integrating with Globus toolkit (a well adopted tool to enable Grid/Cloud computing environment). We also, created simulation experimental results to capture the economic and time efficiency of the proposed solution.

Keywords

Grid/Cloud computing Scheduling, Decentralized Environment, Combinatorial Auction, Grid/Cloud Bidding Language, Winner determination, Grid/Cloud Privacy Model.

Dedication

To my parents Dr. Omar Aburukba and Mrs. Tharwat Almadhoun, who gave me everything, including their love, inspiration, support, and heartfelt prayers.

Acknowledgments

First and foremost, I would like to thank and praise *Allah* almighty, the only One *God*, for the opportunities and support I was given along the way. This work would not have been possible otherwise.

I would like to express my gratitude to my supervisor Professor Hamada Ghenniwa for his encouragement, guidance, and support. I value his respect towards professionalism and the desire to excel higher and higher. He has made me competent in organizing and managing research activities and projects within the CDS Group and in applying them within the industry environment.

I wish to express my gratitude to my co-supervisor Dr. Weiming Shen for his support and for being there whenever I needed his help and feedback.

I am also very grateful to the members of the examining board for their recommendations and suggestions. Their feedback was quite valuable to the final revisions and editing of this work.

I would also like to acknowledge the support of:

- EK3 technologies for providing me the opportunity to work with their team to apply my research within their domain and environment;
- Natural Sciences and Engineering Research Council of Canada (NSERC) for the financial support over the years;
- Afshan and Adrian for the collaborative work of Privacy and the Grid implementation;
- wise people who inspired my thinking through many discussions and collaborative projects in the Cooperative Distributed Systems Engineering Group (Dr. Abdulmutalib, Dr. Chun, Dr. Yunjiao, Dr. Daisy, Wafa, Mohammed, Sherif) and EK3 technologies (Nick, Ed, Ken, Dennis, David, Weiping, Mark, Justin, Kevin, George, Shawn, Adam).

My wholehearted thanks to my parents (Dr. Omar Aburukba and Mrs. Tharwat Almadhoon), my wife (Raghdah Eldaour), and my children (Marya and Omar) for their continuous love, patience, and encouragement.

Table of Contents

List of Tables

List of Figures

List of Models

Chapter 1

1 Introduction

This chapter introduces the context of the research explored in this thesis. It starts with the fundamental motivations behind decentralized and coordinated organization of Grid/Cloud systems; including resource allocation systems. The chapter thereafter provides discussion on the problem and the issues scope of the work and the outline of the thesis.

1.1 Overview

In the last few years, we have seen the emergence of a new generation of business that operates over the Internet. The Internet has become a medium for organizations, businesses and individuals to collaborate because of technological and economic benefits. The complexity of these networks is increasing given their assets of the sub-networks that provide access to services and resources. These networks serve to strengthen business-customer relationships, increases profitability and customer satisfaction. Grid/Cloud computing paradigm has quickly become to realization. However, the integration of decentralized services and resources over the internet is still a challenge.

In the mid-1990s, the term Grid was coined to describe technologies that would allow consumers to obtain computing power on demand. Ian Foster [Foster et al., 2002] and others proposed that by standardizing the protocols used to request computing power, the creation of a Computing Grid could happen, analogous in form and utility to the electric power grid. Standards organizations (e.g., OGF, OASIS) defined relevant standards. The term was also adopted by industry as a marketing term for clusters. But no viable commercial Grid Computing providers emerged, at least not until recently.

In early 2008 the term “cloud computing” was created. Many definitions exist in the literature about Grid and Cloud computing. However, the vision of both the cloud and the Grid is the same which is to reduce the cost of computing, increase reliability, and increase flexibility by transforming computers from something that we buy and operate

ourselves to something that is operated by a third party [Foster et al., 2008]. We view the "cloud" term as another marketing term hype of the Grid computing as they share the same vision, fundamental characteristics and challenges. A similar view is given by many experts defined in [Geelan, 2009].

Grid/Cloud computing is a computational paradigm that utilizes networked computing systems in which applications or services plug into a "power Grid" or "Internet Cloud" of computation for execution. A Network computing system is a virtual system that is formed by processors and networks that agree to work together by pooling their resources. Grid/Cloud computing is a generalized networked computing system that scales to internet levels and handles data and computation seamlessly.

Traditional computational models include three elements: computational power (processors and memory), storage, and software (services). The overall goal of Grid/Cloud computing is to allow applications to utilize computational power, storage, and services as exchangeable commodities. Utilizing such computational power from multiple sources increases the system throughput.

The Grid/Cloud systems can be classified depending on the type of usage. Similar to traditional computation model, those computation elements are the main elements in the Grid/Cloud system. However, instead of the traditional centralized node that does all the computation, the Grid/Cloud has different nodes that are distributed. The Grid/Cloud computing systems can be classified into:

- Computational: denotes a system that has a high aggregate capacity of distributed processors. It harnesses machines in "cycle-stealing" mode to have higher computational capacity than the capacity of any constituent machine in the system.
- Data: provides an infrastructure for creating information from data repositories such as data warehouses.
- Service: refers to systems that provide services that are not provided by any single local machine. An aggregate of services can compose a new service.

This thesis focuses on the Grid/Cloud systems, where participants have the will to collaborate with others in contributing their resources within the environment. In such setting, users provide their resources to be utilized.

1.2 Scheduling Problem in the Grid/Cloud

This thesis focuses on the scheduling problem within the Grid/Cloud environment. In traditional scheduling, a central decision maker is equipped with all the relevant knowledge of the problem, and would be asked to derive a solution that fulfills all the necessary side constraints, optimizing a global performance criterion. The nature of the Grid/Cloud environment is that decisions are taken by several independent entities and those entities might be aiming at optimizing their own objectives rather than the performance of the system as a whole. Entities in this environment are self-interested and willing to share their resources. Such environment calls for models and techniques that take the strategic behavior of individual units into account, and simultaneously keep an eye on the global performance of the system. Strategic situations are traditionally analyzed in Economic theory. In classical economic theory, there are several market models for specific trading situations and structural behaviors. We view Grid/Cloud environment as a marketplace with several participants whose behavior is bound and determined by a diverse set of specialized services, resources and objectives. Economic theory proposed the use of markets to govern and provide efficient allocation of resources.

The MIT Dictionary of Modern Economics [Pearce, 1986] defines a market as a context in which the sale and purchase of goods and services take place.

The Dictionary of economics [Rutherford, 1992] suggests a definition by which market is a medium of exchanges between buyers and sellers. A good is the economic abstraction for a thing that imparts utility to its possessor or recipient.

[Tucker, 1998], "a market is a medium in which autonomous agents exchange goods under the guidance of price in order to maximize their own utility".

Market-based resource allocation systems rely on consumers to set values on resources that they require. Market mechanism is to provide an allocation that is optimal. The fundamental principle is that resources are priced based on the aggregated supply and demand. Consumers seek a quantity of resource that maximizes their utility given the current market price. Trade occurs at a clearing price that balances supply and demand as shown in Figure 1. Such allocations are economically efficient. This means no reallocation can make one better off without making another worse. Applying the economic-based framework offers an effective way to solve the issues of scheduling problems in the Grid/Cloud environment such as decentralization, autonomy, resource sharing, heterogeneity, and quality of solution.

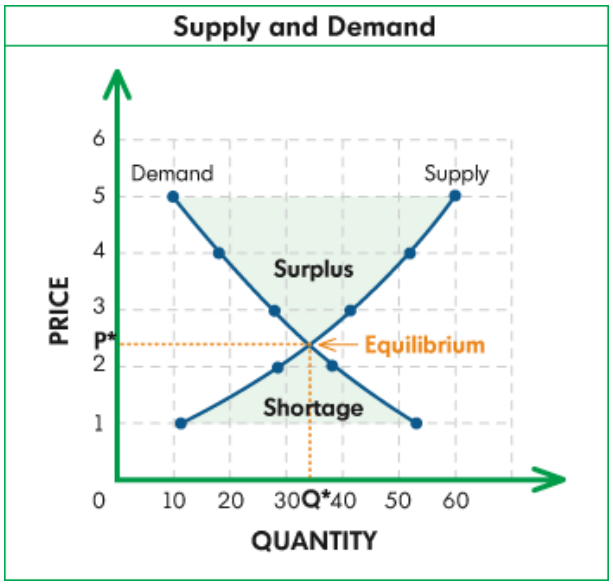

Figure 1: Supply and demand curves and equilibrium point. Image from the economic blog: http://enthusiasm.cozy.org/

1.3 Problem Scope and Issues

In this thesis, we address the challenges related to modelling and developing a practical architectural solution for resource scheduling in the Grid/Cloud environment that supports both economic efficiency and allocation adequacy based on the characteristics of the environment. Moreover, there is an emergent demand for expressive mechanisms in the Grid/Cloud computing environment. For example, the ability to express time and quality as well as co-allocation constraints. It is recognizable that any adoption of auction

mechanisms must support a bidding language with the ability to express complicated valuations over multiple attributes. The design of a bidding language plays a key role in the allocation problem, preference elicitation and winner-determination [Lehmann et. al. 2006]. A well-known expressive mechanism is a combinatorial auction (CA) [Benisch et. al. 2008][Lubin et. al., 2008], which allows participants to express valuations over bundles of items. In this thesis, we develop a tree-based requirement specification language (TBRSL) that allows bidders to directly express their requirements, such as time boundaries, resource requirement specification, and valuations. The proposed bidding language addresses challenges related to expressiveness as the ability to specify any set of preferences in the Grid/Cloud; ease-of-use as the ability to express structured preferences directly; computational-efficiency as the ability to support computationally tractable winner-determination algorithms. In addition to the computational efficiency, we address other attributes that are essential to the winner determination mechanism in the Grid/Cloud. Such attributes are: allocative efficiency, strategy-proofness, and individual rationality.

Moreover, in an open environment such as the Grid/Cloud, it is inadequate to assume that entities consider privacy of information. It is essential that entities receive privacy protection in order to safely coordinate with each other. The work in [Samani et. al., 2012] identifies the elements of privacy situations and proposes a risk assessment model for evaluating the risk of interactions of two entities. This includes elements such as trust level, severity of operation on information, negotiated agreement between entities, relevancy of the type of the requested information and the type of the offered service, sensitivity, cost and criticality of information and the information gain of exposing the information to other entities. The risk assessment model considers all these elements and calculates the risk of privacy violation in a specific interaction [Samani et. al., 2012]. Utilizing the risk assessment procedure facilitates quantifying privacy interactions. It can lead to evaluate privacy interactions in terms of Privacy Protection Level (PPL). In this thesis, we provide a scheduling solution given privacy concerns requirements. We analyze the privacy concerns to be applied to the Grid/Cloud computing scheduling problem and utilize the proposed solution to the bidding language proposed in Chapter 5

and the winner determination mechanism proposed in Chapter 6 within the solution for the scheduling problem given privacy concerns.

1.4 Outline of the Thesis

The rest of the thesis is structured as follows. Chapter 2 reviews scheduling problem models and related solution approaches for the Grid/Cloud environment. Chapter 3 presents an overview of the Grid/Cloud computing system. Chapter 4 analyzes the scheduling problem in the Grid/Cloud and formulates models based on the completion time of consumers and resource utilization or providers and describes the mapping of the scheduling problem in the Grid/Cloud to economic based models. Chapter 5 describes the proposed Grid/Cloud based bidding language. Chapter 6 proposes a winner determination algorithm for the Grid/Cloud scheduling problem. Chapter 7 presents the implementation architecture, integration with Globus, and results validation. Chapter 8 provides a brief conclusion.

Chapter 2

2 Related Work

Effective scheduling is a key challenge for performance and quality driven requirements of Grid/Cloud computing requests. Scheduling is a process of finding the capable resources that can execute the Grid/Cloud requests (tasks) at specific times that satisfy specific performance quality measure such as execution time minimization, as specified by Grid/Cloud users. In this chapter, we review scheduling algorithms, techniques, and frameworks used for scheduling tasks on the Grid/Cloud.

2.1 Scheduling Structures Overview

The architecture of a scheduling infrastructure is very important with regards to scalability, autonomy, and performance of the system [Hamscher, 2000]. It can be divided into three categories: centralized, distributed and decentralized.

In a centralized scheduling architecture [Yu and Buyya, 2009], scheduling decisions are made by a central controller for all the tasks. The scheduler maintains all information about the tasks and keeps track of all available resources in the system. Centralized scheduling organization is simple to implement and easy to deploy. However, it is not adequate for the Grid/Cloud because of the nature of the Grid/Cloud computing environment.

In distributed scheduling, there is a central manager and multiple lower-level entities. This central manager is responsible for handling the complete execution of a task and assigning the individual tasks to the low-level providers. Each lower-level entity scheduler is responsible for mapping the individual tasks into Grid/Cloud resources. Such approaches are not adequate since it requires entities to deploy different scheduling policies to the central manager [Hamscher, 2000]. The failure of the central manager results in entire system failure.

In contrast, decentralized scheduler [Ranjan et. al., 2008] negates the limitations of centralized or distributed structures with respect to fault-tolerance, scalability, autonomy,

and most importantly the adequacy for the Grid/Cloud computing environment as it will be analyzed in Chapter 3. A decentralized scheduling approach assumes that each entity is autonomous and has its own control that derives its scheduling decision based on its policies. However, if the decisions are taken by several independent units, it might be the case that these units aim at optimizing their own objectives rather than the performance of the system as a whole. Such situations call for models and techniques that take the strategic behavior of individual units into account, and simultaneously keep an eye on the global performance of the system. Strategic situations are traditionally analyzed in Game Theory as well as certain areas of Economic Theory.

2.2 Scheduling Objective

Generally, schedulers generate the mapping of tasks to resources based on some particular objectives. Schedulers employ a function that takes into account the necessary objectives to optimize a specific outcome. The commonly used scheduling objectives in a Grid/Cloud computing environment are related to the tasks completion time and resource utilization.

The scheduler uses a specific strategy for mapping the tasks to suitable Grid/Cloud resources in order to satisfy user requirements. However, the majority of these scheduling strategies are static in nature [Topcuoglu et al. 2002]. They produce a good schedule given the current state of Grid/Cloud resources and do not take into account changes in resource availability. On the other hand, dynamic scheduling [Rahman et. al., 2007] considers the current state of the system. It is adaptive in nature and able to generate efficient schedules, which eventually minimizes the completion time of tasks as well as improves the overall performance of the system.

2.3 Entities Coordination in the Grid/Cloud

Entities in the Grid/Cloud are viewed as independent entities that are able to perform some functionality and have their own will in sharing their capabilities. The challenge with such systems is how to manage the interdependencies among the entities having no global control. The effectiveness of managing interdependencies of entities in the

Grid/Cloud depends on the coordination among different entities in the environment. Lack of coordination results in communication overhead and eventually reduces performance of the system. The process of coordination with respect to application/service scheduling and resource management in the Grid/Cloud involves dynamic information exchange between various entities in the system.

2.3.1 Coordination Mechanism

Coordination mechanism reduces and resolves the problems associated with interdependencies. Hence, a coordination mechanism contains a set of decision points (coordinated-control) and interaction protocols directed to deal with the interdependency problems. Interaction protocols are the mean by which an entity interacts with another entity through some communication protocol. Effective coordination amongst entities in the Grid/Cloud requires adequate coordination mechanisms and negotiation policies. Market-based coordination mechanisms are well adopted in the Grid/Cloud environment.

A Market based mechanism views the Grid/Cloud computing environment as a virtual marketplace in which economic entities interact with each other through buying and selling computation, storage resources, and services. Such a coordination mechanism is used to facilitate efficient resource allocation. In such mechanism, the resource provider works as a manager that exports its local resources to contractors, and resource brokers are responsible for decision regarding admission control based on negotiated Service Level Agreements (SLA).

2.3.2 Coordination Structure

Coordination structure is the pattern of decision making and communication that are required while resolving problems associated with interdependencies between entities.

The interaction among entities is coordinated by the utilization of some particular communication devices that can be divided into two types: One-to-one and One-to-many. One-to-many broadcast communication is simple but very expensive in terms of the number of messages and network bandwidth usage. This overhead can be drastically

reduced by adopting One-to-one among the resource providers and consumers through establishment of a Service Level Agreement.

2.4 Economic-Based Approaches Background

The economic approaches are based on microeconomic theories, particularly general equilibrium theory and mechanism design. The economic based approaches take the assumption that agents chose their own strategies. In other words, agents have control of their own behavior. In microeconomics, there are two approaches to modeling agent behavior:

1. **Price-taking/competitive equilibrium:** In this model the equilibrium state is defined by the condition that an agent plays a best-response to the current price and allocation in the market, without modeling either the strategies of other agents or the effect of its own actions on the future state of the market.
2. **Game-theoretic/mechanism design:** In this model the equilibrium state is defined by the condition that agents play a best-response strategy to each other and cannot benefit from a unilateral deviation to an alternative strategy.

Mechanism design theory and game-theoretic modeling is most relevant when one or both of the following conditions hold:

- the equilibrium solution concept makes weak game-theoretic assumptions about agent behavior, such as when a mechanism can be designed with a dominant strategy equilibrium, in which agents have a single strategy that is always optimal whatever the strategies and preferences of other agents; or
- there are a small number of agents and it is reasonable to expect agents to be rational and well-informed about the likely preferences of other agents.

Competitive equilibrium theory and price-taking modeling is most relevant in:

- large systems in which the effect of an agent's own strategy on the state of a market is small, or
- when there is considerable uncertainty about agent preferences and behaviors and no useful mechanism with a dominant strategy equilibrium.

Hence, competitive equilibrium approaches is more relevant for the Grid/Cloud systems given its nature. In the next sections, we review the different approaches within the competitive equilibrium theory.

2.5 Price-Taking/Competitive Equilibrium Approaches

Markets provide a level of abstraction based on which desirable global effect can be achieved, such as fair allocation of resources, through coordination, i.e. buying and selling, among individual agents. Market-based approaches can provide several advantages [Wellman et al., 2001]. Markets are naturally decentralized. This means that agents in the market have their own knowledge and control where agents are capable to making decisions about how to bid based on the prices and their own relative valuations of the goods. The bids and valuations reflect the agent's strategies to achieve its goal. This implies that agents are autonomous and rational in its decision whenever it is feasible. Communication is limited to exchange decisions (bids and prices) between agents. Negotiation mechanisms can elicit the information necessary to achieve Pareto and global optimal. Pareto optimal solution implements outcomes for which no alternative outcome is strongly preferred by at least one agent, and weakly preferred by all other agents.

Several economic models that support distributed rational decision making have been studied in [Sandholm, 1999]. Some of them, including general equilibrium market mechanisms, and auctions. In the rest of this section, we review these models.

2.5.1 General Equilibrium Market Mechanisms

In economics, the concept of a set of interrelated goods in balance is called general equilibrium [Wellman, 1993]. General equilibrium theory provides a distributed method for efficiently allocating goods and resources among agents based on market prices. This model assumes agent behaviour as price-taking or myopic best-response. The equilibrium state is defined by the condition that an agent plays a best-response to the current price and allocation in the market, without modeling either the strategies of other agents or the effect of its own actions on the future state of the market. The model is most relevant in large systems in which the effect of an agent's own strategy on the state of a market is

small, or when there is considerable uncertainty about agent preferences and behaviors and no useful mechanism with dominant strategy equilibrium. In other words, producers are sharing their goods (such as capabilities) by putting specific price values based on their strategy, and consumers must have the goods from the producers to achieve their goals.

One of the first general equilibrium based approaches is called *market-oriented programming* (MOP) [Wellman, 1993]. In MOP, agent activities are defined in terms of resources required and produced, reducing an agent's decision problem to evaluating the tradeoffs of acquiring different resources [Wellman et. al., 2001]. These tradeoffs are represented in terms of market prices, which define a common scale of value across the various resources. The problem for designers of computational markets is to specify the configuration of resources traded, and the mechanism by which agent interactions determine prices. The advantages of utilizing market approaches for decentralized scheduling problems are:

- Markets are naturally decentralized. Agents make their own decisions about how to bid based on the prices and their own relative valuations of the goods.
- Communication is limited to the exchange of bids and prices between agents and the market mechanism. In particular settings, it can be shown that price systems minimize the dimensionality of messages required to determine Pareto optimal allocations.
- In some well-characterized situations, some mechanisms can elicit the information necessary to achieve Pareto and global optima.

2.5.2 Commodity Market

In a commodity market various suppliers and consumers register in the commodity market. Each participant decides upon a course of action, which may consist of the sale of some commodities and the purchase of others. Thus supply and demand functions for each commodity can be defined as the aggregate behavior of all participants. These are determined by the set of market prices for the various commodities. Equilibrium for the economy is established when supply is equal to demand (i.e., the excess demand function

has a zero value). Practically, it will be sufficient to find approximate equilibrium in the sense of finding a price that makes the values of the excess demands close to zero.

The commodity market governs the trading behavior of the participant entities in the session. The market recognizes three types of entities, namely, market-mediators, consumers, and suppliers. Each market session is assigned to a mediator to coordinate the actions taken by consumers and suppliers in a way that will eventually clear its respective market. There is a one-to-one correspondence between market mediators and commodities. Initially, a mediator is assigned to a specific commodity market and broadcasts a randomly chosen initial price vector to all registered participants in its market. Then, each participant computes the demand function for each of its commodities of interest. Each demand function specifies the net quantity demanded of a commodity (which for a net supply is negative) as a function of its price, assuming that the prices for the remaining commodities are constant. The mediator, upon receiving the demand and supply from all participants, computes the clearing price, for which the aggregate excess demand is zero. The mediator then notifies the participants of the new price. Upon seeing new prices, the consumers and suppliers compute revised demand functions as necessary based on these new prices. This process continues until the prices' changes are within a specified threshold. Then the process terminates and the mediator reports the final state of the price vector as the equilibrium.

2.5.3 Auction Market

The three key players involved in auctions are: resource owners (providers), auctioneers (mediators), and buyers (consumers). The auctioneer sets the rules of auction which is agreed by both consumers and the providers. Auctions basically use market forces to negotiate a clearing price for the service. Usually auctions are used particularly for selling goods/items within a set duration. Auctions can be classified into two types, single auctions and double auctions.

The single auction model supports one-to-many negotiation, between a provider (seller) and consumers (buyers), and reduces negotiation to a single value (i.e. price). The types

of auctions related to the one-to-many negotiation are: English Auction, First-price sealed-bid auction, Vickrey auction, and Dutch auction.

In the double auction model, buyers (bids) and sellers (asks) may be submitted at anytime during the trading period. If at any time there are open bids and asks that match or are compatible in terms of price and requirements (e.g., quantity of goods or shares), a trade is executed immediately. In this auction orders are ranked highest to lowest to generate demand and supply profiles. From the profiles, the maximum quantity exchanged can be determined by matching *asks* (starting with lowest price and moving up) with demand bids (starting with highest price and moving down). All auctions can be classified as open or closed (sealed) auctions.

Closed Auction

The closed auction uses the direct-revelation principle which states that it is sufficient to restrict attention to incentive compatible mechanisms related to collecting bids from participants only once. In a single-bid mechanism each agent is simultaneously asked to report its valuation. In an incentive-compatible (IC) mechanism each agent finds it in their own best interest to report its valuation truthfully. The mechanism design problem defines functions that map valuations to outcomes, subject to constraints that ensure that the mechanism is incentive-compatible.

The single-bid mechanism does not imply that incentive-compatibility is given. The single-bid principle conditions that if a particular set of properties can be implemented in the equilibrium of some mechanism, then the properties can be implemented in an incentive-compatible mechanism. On the other hand, the single-bid principle ignores computation and communication complexity.

Open Auction

The open auction type uses the indirect-revelation principle. The principle is based on mechanisms, in which agents are not required to submit (and compute) complete and exact information about their private valuations Indirect mechanisms, such as those based on prices, also go some way to distributing the calculation of the outcome of a

mechanism across agents rather than requiring the mechanism infrastructure (such as the auctioneer) to compute the winners and the payments.

An example of indirect mechanisms include ascending-price auctions, in which agents submit bids in response to prices and the auctioneer maintains a provisional allocation and adjusts prices. For example, the English auction is an ascending-price auction for a single item in which the price increases until there is only one bidder left in the auction.

For the open auction market to happen there must be at least two agents in the bidding process to make progress towards the outcome and agents can follow the equilibrium strategies.

2.5.3.1 Auction Mechanisms

Vickrey Auction

The GVA is a sealed bid auction. Each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of the second highest bidder [Sandholm et al., 2005]. The dominant strategy in Vickrey is for bidders to report its true valuation function.

The auctioneer agent

- Calculates the allocation (x_i^*) that maximizes the sum of the bids subject to the items constraint.
- Calculates the allocation $(x_{\sim i}^*)$ that maximizes the sum of the bids other than that of bidder agent *i* such that it excludes all items allocated to agent *i*.
- Announces the winners and their payment given by $p_i = \sum_{j \neq i} v_j(x_{\sim i}^*) - \sum_{j \neq i} v_j(x^*)$.

Under the assumption of quasilinear preferences, each bidder agent calculates its utility. For bidder agent *i* the utility will be

$$u_i(x^*) = v_i(x^*) - p_i = v_i(x^*) - \sum_{j \neq i} v_j(x_{\sim i}^*) - \sum_{j \neq i} v_j(x^*).$$

First-price sealed-bid auction

This auction is a single-bid type where each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of his bid. The best strategy is bid less than its true valuation and it might still win the bid, but it all depends on what others bid.

Call Market

Call market is a double auction type of market in which each transaction takes place at predetermined intervals and where all of the bids and asks are aggregated and handled at once. The exchange determines the market clearing price based on the number of bids and asks. In call market, orders are filled as soon as a buyer/seller is found for any given order at an agreed price.

English Auction

This auction is an outcry type where all bidders are free to increase their bids exceeding other offers. When none of the bidders are willing to raise the price anymore, the auction ends, and the highest bidder wins the item at the price of his bid.

The dominant strategy for English auction is to always bid a small amount "higher" than the current highest bid, and stop when its private value price is reached. In correlated value auctions, the policies are different and allow the auctioneer to increase the price a constant rate or at a rate the entity wishes. Entities that are not interested in bidding anymore can openly declare so (open-exit) without re-entry possibility. This information helps other bidders and gives a chance to adjust their valuation.

Dutch Auction

This auction is an outcry type where the auctioneer starts with a high bid/price and continuously lowers the price until one of the bidders takes the item at the current price or a predetermined reserve price (the seller's minimum acceptable price) is reached. The winning participant pays the last announced price.

Continuous Double Auction

Continuous Double Auction allows for many buyers and sellers to continuously submit bids for the purchase and sale of a commodity.

Iterative Bundle Auction

The iterative bundle auction has the similar strategy as the GVA mechanism with the exception of allowing iteration. Iterative bundle auctions are indirect implementations of GVA [Parkes and Ungar, 2000, Parkes and Kalagnanam, 2005, Bikhchandani and Ostroy, 2006]. This class of auction has practical significance change in the agents behaviour from GVA since it allows agents to reveal their preference information as necessary as the auction proceeds, and agents are not required to submit (and compute) complete and exact information about their private valuations.

Agents can use bundle bids to directly express contingent demands for items. However, a direct implementation of GVA cause prohibit computation and communication cost. To avoid this, indirect implementations of GVA have been proposed. This class of auction, called *iBundle*, has practical significance because it addresses the computational and informational complexity of bundle auctions and allows a tradeoff between performance and computation.

Sequential and Simultaneous Auctions

Sequential and simultaneous auctions price bundles as the sum price of the individual items. Sequential auctions suppose that the set of resources of interest are auctioned in sequence. Agents bid for resources in a specific, known order, and can choose how much (and whether) to bid for a resource depending on past successes, failures, and prices. Sequential auctions are particularly useful in situations where setting up a combinatorial or simultaneous auctions are infeasible.

Simultaneous auctions sell multiple goods in separate markets simultaneously. Agents have to interact with simultaneous but distinct markets in order to obtain a combination of resources sufficient to accomplish their task. Real-world markets typically operate

separately and concurrently despite significant interactions in preferences or costs [Wellman et al., 2004].

2.6 Grid/Cloud Scheduling Approaches

2.6.1 Economic-based Scheduling Approach

Mechanisms inspired in economic principles come from observing of how economies allocate resources. The work in [Nakai et al., 2003] made a critical analysis of the General Equilibrium theory and the applicability of markets to global scheduling in Grid/Cloud. Their conclusion is that General Equilibrium fails due to the perfect competition that drives an economy. Certainly, competition in a market does not lead to finding an equilibrium solution. In other words, the optimal scheduling solution can not be reached when entities do not cooperate. For that reason mechanism design has been studied to enable entities to participate cooperatively in a market.

Market-based models for resource allocation can bring benefits to Grid/Cloud infrastructures. The work in [Shneidman et al., 2005] points out that many computer systems have reached a level where the goal is not always to maximize utilization; instead, when demand exceeds supply and not all needs can be met, a policy for making resource allocation decisions is required. Hence, market-based approaches are a good choice to carry out policy-directed resource allocation. It is natural to consider mechanisms based on economic principles for the Grid/Cloud because it comprises multiple entities, established by different communities that are heterogeneous in terms of goals, priorities and quality of service requirements.

OurGrid [Andrade et al., 2003]: is a resource sharing system organized as a P2P network of sites that share resources fairly forming a Grid to which they all have access. OurGrid provides connected sites with access to the Grid resources with the minimal guarantees needed. OurGrid supports the execution of Bag-of-Tasks (BoT) applications; parallel applications composed of a set of independent tasks that do not communicate with one another during their execution. OurGrid does not require offline negotiations if a resource owner wants to offer their resources to the Grid. The three participants in OurGrid's protocol: clients, consumers, and providers. A client requires access to the Grid resources

to run their applications, the consumer receives requests for resources from clients, proceeds to find the resources able to serve the request, and then executes the tasks on the resources, and the provider manages the resources shared in the community and makes them available to consumers. OurGrid uses a resource exchange mechanism termed network of favors. A participant *A* is doing a favor for participant *B* when *A* allows *B* to use *A*'s resources. According to the network of favors, every participant does favors for other participants expecting the favors to be reciprocated. In conflicting situations, participants prioritize those who have done them favors in the past. The more favors participants do, the more rewards they expect. The participants account locally for their favors, and cannot profit from them other than expecting other participants to do favors for them in return. Experiments demonstrated that the mechanism performs more fairly when the network is large. This approach does not support other Grid/Cloud characteristics such as QoS. Moreover, tit-for-tat mechanism is expensive with respect to communication between entities in a distributed system.

Nimrod-G: [Buyya et al.,2000a] [Buyya et al.,2000b] is a Grid resource broker that allows managing and routing task applications on computation Grids. It employed the commodity market for resource management and scheduling. Several algorithms called deadline and budget constrained (DBC) scheduling algorithms are presented which consider the cost and makespan of a job simultaneously. These algorithms implement different strategies. For example, guaranteeing the deadline and minimizing the cost or guaranteeing the budget and minimizing the completion time. The difficulties to optimize these two parameters in an algorithm lie in the fact that the units for cost and time are different, and these two goals usually have conflicts (for example, high performance resources are usually expensive).

The Time Optimization scheduling algorithm attempts to complete as quickly as possible, within the available budget. The algorithm initially considers the next available completion time given the current assigned jobs. The resources are sorted by the next completion time and then one job is assigned to the first resource for which the cost per job is less than or equal to the job budget.

The Cost Optimization scheduling algorithm attempts to find a schedule as economically as possible within the deadline. The algorithm sorts the resources by increasing cost, then for each resource assign jobs to the resources without exceeding the deadline.

The Conservative Time Optimization scheduling algorithm attempts to complete the schedule as quickly as possible within specific budget constraint. It ensures that a minimum of "the budget-per-job" from the total budget is available for each unprocessed job. The algorithm splits a resource by whether the cost per job is less than or equal to the budget per job. Then for the cheaper resources, assign jobs in inverse proportion to the job completion time (e.g. a resource with completion time = 5 gets twice as many jobs as a resource with completion time = 10).

The work experiments with the commodity market. We believe market approaches is a suitable approach, however, with it comes other challenges that need to be addressed such as communication, strategic, and winner determination complexities. Entities in the Grid are autonomous. Market mechanisms provide a way for entities to coordinate, however, they are not necessarily cooperating. The reason we need entities to cooperate is that by doing so we are guaranteed to find a pareto optimal solution in the decentralized environment. Otherwise, a market mechanism is not guaranteed to work effectively.

[Ernemann et al., 2005] proposed a scheduling model that is not restricted to a single central scheduling instance. Each domain can act independently and may have individual objective policies. Also, each task request can include an individual objective function. They defined a description language to formulate objective functions that are then evaluated to scalar values at run time. The scheduling system combines the different objective functions to find the equilibrium between supply and demand. The work used two heuristics: one to fix a job size and another to estimate start times. They divide the job into several smaller parts as specified using two parameters, the minimum and maximum number of resources a job part may be allowed to use. The second heuristic estimates the start times for the entire job. All job parts must be executed at the same time, but the initiating scheduler may have only limited information about the schedules

on the other resources. The work used the commodity market model. The result of this work showed that the economic scheduling outperforms the conventional first-come first-served strategy.

This approach utilizes commodity market and specified a description language for users to describe their requirement to achieve specified criterions. In a decentralized environment criterions are not common between entities and we believe that a bidding language is required to gather entities requirements, yet does not interfere with the entities' private information such as entities' objectives. A mechanism is also required to induce entities not to miss represent their requirements. We believe more studies need to be conducted to find the suitability of the approach given the other types of markets instead of comparing to the first-come first-served strategy.

[Young et. al 2003] compared game theory approach (static game of complete information) with the simulated annealing under the criteria of time and cost optimization. The proposed game theory algorithm uses a list structure and iterates through every single strategy within the list without having a specific search heuristic. Their results show that the simulated annealing approach achieved better quality than game theory approach. Their claim is that game theory approach has proved disappointing, being outperformed by simulated annealing approach. This is due to the implementation limitation of uncooperative game theory.

Those two approaches are not suitable since both approaches require entities to provide information to the center that provides the scheduling solution. The Grid environment on the other hand, is decentralized and entities are autonomous. Moreover, they focused on non-cooperative entities. As mentioned earlier if entities do not cooperate, the outcome of the scheduling solution can be far away from the optimal.

[Wang et al., 2007] This work presented an auction-based winner determination formulation and algorithm for the decentralized scheduling problem. The work used the mathematical modeling of the winner determination. The proposed approach consists of an iterative bidding protocol, requirement-based bidding languages, and a constraint-based winner determination approach. The proposed requirement-based bidding language

allow bidders to bid for the processing of a set of jobs with constraints by imposing a time window discretization on resources. The winner determination algorithm uses a depth first branch and bound search. Also, the work used a constraint directed scheduling procedure at each node to verify the feasibility of the allocation. The work employed an experiment against the commercial optimization engine CPLEX 10.0 and showed that the proposed algorithm is faster on average over a set of winner determination problems of decentralized scheduling generated based on job shop constraint satisfaction benchmark problems.

2.6.2 Heuristics

Since the Grid/Cloud computing scheduling is an NP-hard problem, we rely on heuristic based strategies to achieve near optimal solutions within polynomial time. The following subsections present some of the well-known heuristics for scheduling.

Min-Min

This approach prioritizes tasks and generates a schedule based on the priority. This priority is generated based on the task's Expected Completion Time on a resource. The approach arranges the tasks into several independent tasks groups. Those groups are then scheduled iteratively. Every iteration takes the set of unmapped independent tasks and generates the Minimum Expected Completion Times (MECT) for each task. The task that has the smallest MECT value over all tasks is selected to be scheduled first at this iteration to the corresponding resource. This continues until all tasks are scheduled. This approach was proposed by Maheswaran et al. [Maheswaran et. al.,1999] and has been employed for scheduling tasks in Grid projects such as vGrADS [Blythe et. al., 2005] and Pegasus [Mandal et. al., 2005].

Max-Min

This approach is similar to the Min-Min approach, however, Max-Min sets the priority to the task that requires the longest execution time. Every iteration takes the set of unmapped independent tasks and generates the Maximum Expected Completion Times (MECT) for each task. The expectation is to complete the task at the earliest time by

assigning longer tasks to comparatively best resources. The approach is proposed in [Maheswaran et. al.,1999] and [Mandal et. al., 2005]

HEFT

Heterogeneous Earliest Finish Time (HEFT) [Topcuoglu et al. 2002] gives higher priority to the tasks having higher rank value. The rank value is calculated by utilizing average execution time for each task and average communication time between resources of two successive tasks, where the tasks in Critical Path get comparatively higher rank values. Then it sorts the tasks by decreasing order of their rank values and the task with higher rank value is given higher priority. In the resource selection phase, tasks are scheduled based on their priorities. Each task is assigned to the resource that can complete the task at the earliest time. This approach considers the entire workflow tasks rather than unmapped independent tasks. This approach was used by [Topcuoglu et al. 2002] [Wieczorek et. al, 2005] [Fahringer et. al., 2005].

2.6.3 Other Scheduling Approaches in the Grid/Cloud

Condor-G: Condor-G [Frey et al., 2001] employs components from Globus [TGA, 2013] and Condor [Wright, 2003] to allow users to utilize resources spanning multiple domains as if they all belong to one personal domain. Condor-G uses Condor mechanisms to match locally queued jobs to the resources advertised in a FIFO strategy without any long-term optimization.

Sun Grid Engine (SGE) [Bulhoes et al., 2004]**:** a resource management and scheduling system from Sun Microsystems that is used to optimize the utilization of software and hardware resources. Tasks submitted to the master node in and SGE cluster are held in a spooling area until the scheduler determines that the task is ready to run. SGE matches the available processors/resources to a task's requirements such as, available memory, CPU speed, which are periodically collected by the execution node. Once a processor/resource becomes available for execution of a new task, SGE dispatches the task with the highest priority and matching the requirements. SGE uses two sets of criteria to schedule tasks: task priorities, and equal share.

The task priority criterion concerns the order of the scheduling of different tasks, a first-in-first-out (FIFO) rule is applied by default. All pending tasks are inserted in a list, with the first submitted task being at the head of the list, followed by the second submitted task and so on. The FIFO rule sometimes leads to issues, especially when a series of tasks are submitted at almost the same time. All the tasks that are submitted in this case are assigned within the same queue and have to potentially wait a very long time before execution.

The Portable Batch System (PBS) [Li and Baker, 2005]: a resource management and scheduling system in a cluster-based computing environment. PBS uses a master node, and an arbitrary number of execution and tasks submission nodes. The master node is the central manager of a PBS cluster. PBS supports the following constraints of the tasks:

- Tasks can be sequential or individual tasks.
- Tasks can have a list of required processors (speed, capabilities)
- Tasks can have priority constraints
- Tasks can have a duration for execution
- Tasks can have dependencies with other tasks
- Tasks can be suspended and later resumed

Jobs submitted to PBS are put in job queues. Two main queue types are defined: routing and execution queues. Jobs in the execution queue are candidates for execution. Jobs in the routing queue are candidates for routing to a new destination.

2.7 Privacy in the Grid/Cloud

Privacy is a subjective concept and would be treated differently within entities in the Grid [Dey et. al., 2002]. Privacy is a concept that has a major focus in several fields of research. However, because of the subjective nature of privacy, it is difficult to define it. It varies from one perspective to another and from one context to another. There are several theories in privacy such as “the right to be left alone”, “limited access to self” and control over personal information [Solove, 2008]. However, in the Grid/Cloud environment, privacy is typically addressed in the context of “information privacy”. The focus of information privacy is on the operation that is applied on information. It can be

categorized as information collection, information processing and information dissemination. One of the challenges in information privacy is identification which is applying any operation that relates sensitive information to entities [Schwartz and Solove, 2011]. Information or attributes can be classified based on their ability to identify entities. There are attributes that are identifiers to entities such as SIN numbers, personal number and identification information in scheduling tasks and resources. There are also attributes that can be used in combination with others to identify an entity; for example, combination of date of birth, gender, name and zip code. In another example, combination of attributes such as computer design, processor type, vendor and delivering site of a computer can identify super computers. The attributes that directly identify the entities are called "identified" and the attributes that can result in identifying an entity are called "Personally Identifiable Information" (PII). The challenge is that due to improving technology and information processing by which the non PII attributes can be converted to PII attributes, it becomes not possible to directly identify the personally identifiable information [Schwartz and Solove, 2011].

Among the approaches for resolving PII complications, there are rule-based and standard-based approaches. In the context of PII, rule-based approaches are not sufficiently effective. Usually, the rule-based approaches are convenient when the area of social and technological development have reached a fairly stable state [Schwartz and Solove, 2011]. Therefore, in the setting of the Grid/Cloud, a standard (architectural) based privacy management system is required.

Considering the non-clear barrier between PII and non PII information, there are approaches to resolve the PII problem.

- *Reduction:* focuses on "identified" attributes and concerns only with information about identified entities. The "identifiable" concept has been eliminated from this approach [Schwartz and Solove, 2011].
- *Expansion:* In this approach, the identifiable information is considered as critical as identified information. However, from the practical point of view, almost any kind of information can be attributed to an identity. This approach treats the identified and identifiable information equally. This can be considered flawed [Schwartz and Solove, 2011].

- *PII 2.0*: It has been observed that not all of identifiable information has the same risk level of privacy violation. This introduces the concept of risk of revealing information. If the risk of a set of identifiable information is high, then information should not be disclosed [Schwartz and Solove, 2011]. Based on the possibility of conversion of non-PII to PII class and similarly for identifiable information to be converted to identify information, a dynamic risk re-evaluation becomes essential.

Considering the existing scheduling solutions in the Grid/Cloud, attending to privacy issues is lacking. There have been attempts to resolve privacy concerns in DCOP (Distributed Constraint Optimization Problem) [Greenstadt, 2008][Greenstadt et. al., 2006]. DCOP consists of entities that set and control valuation of variables. Entities decide which valuation of the variables has more benefit for them. However, the setting of the problem is based on the assumption that all entities are aware of the constraints of other entities and only the valuation of variables is the private information [Greenstadt et. al., 2006]. Moreover, there is no matching process between what they need and what is offered [Greenstadt, 2008][Greenstadt et. al., 2006]. In contrary, the context of scheduling problem in the Grid/Cloud contains providers that have capabilities and consumers that have requirements. Entities in this configuration are not willing to share their constraints. Therefore, the solutions in DCOP are not fully compatible with the setting of scheduling problem in the Grid/Cloud and are designed for less complicated configurations. Additionally, privacy solutions in DCOP are from an information theoretic perspective [Greenstadt et. al., 2006]. They can be categorized as utility-trade off solutions for privacy [Such et. al., 2012]. For confronting privacy issues in the Grid/Cloud, considering information gain is necessary. However, the social aspects of relationships between entities have a significant role in evaluating privacy [Such et. al., 2012][Dey et. al., 2002].

Chapter 3

3 Grid/Cloud Computing System

Grid/Cloud computing is a computational paradigm that utilizes networked computing systems in which applications plug into a "power Grid" of computation for execution. A Network computing system is a virtual system that is formed by processors and networks that agree to work together by pooling their resources. Grid/Cloud computing is a generalized networked computing system that scales to internet levels and handle data and computation seamlessly.

The traditional computational model includes three elements: computational power (processors and memory), storage, and software (services). The overall goal of Grid/Cloud computing is to allow applications to utilize computational power, storage, and services as exchangeable commodities. Utilizing such computational power from multiple sources increases the system throughput.

The Grid/Cloud systems can be classified depending on the type of usage. Similar to the traditional computation model, those computational elements are the main elements in the Grid/Cloud system. However, instead of the traditional centralized node that does all the computation, the Grid/Cloud has the elements distributed among different nodes. We can classify the Grid/Cloud computing systems as:

- **Computation:** denotes a system that has a high aggregate capacity of distributed processors. It harnesses machines in "cycle-stealing" mode to have higher computational capacity than the capacity of any constituent machine in the system.
- **Data:** provides an infrastructure for creating information from data repositories such as data warehouses. Applications for these systems would be special purpose data mining that correlates information from multiple different high volume data sources
- **Service:** refers to systems that provide services that are not provided by any single local machine. An aggregate of services can compose a new service.

3.1 The Grid/Cloud System: High-Level View

The Grid/Cloud is made up of a number of components that expose computation, storage, and services to the network. A layered logical architecture of the Grid/Cloud is shown in Figure 2b and in relationship to the Internet Protocol architecture Figure 2a. The Grid/Cloud logical architecture in Figure 2b includes additional protocols and services that are built on the Internet protocols and services to support the creation and use of computation and data-enriched environments. Any resource that is on the Grid/Cloud is also, by definition, on the Net. The Grid/Cloud layers as shown in Figure 2b are:

- **Fabric:** Traditionally in the internet architecture, the link layer connects different computation nodes together through different types of mediums such as physical media which includes coaxial cable, and copper wire. The Fabric layer in the Grid/Cloud architecture consists of distributed processors, storage resources that utilize the link layer and are connected by high-bandwidth networks. Each processor runs system software such as operating systems, resource management systems, and relational database management systems. With this mapping, logically, we move traditional computing from being done from the node to being done at the network level.
- **Resource and Connectivity Protocols:** consists of protocols that are built on the core communication protocol (TCP/IP) and used to query entities in the Grid/Cloud Fabric layer and to conduct collaboration between them. Cryptographic protocols allow verification of users' identities and ensure security and integrity of transferred data. These security mechanisms form part of the Grid/Cloud Security Infrastructure (GSI) [Foster et al. 1998]. This layer defines core communication and authentication protocols required for the Grid/Cloud transactions. Communication protocols enable the exchange of data between Fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources.
- **Collective Services:** This includes service monitoring and discovery such as the Brokering service, Monitoring and Diagnostic services for managing and scheduling applications for execution on the processors and resources in the Grid/Cloud.

- **User Applications:** specific services that cater to users by invoking services provided by the layers below and customizing them to suit the target domains.

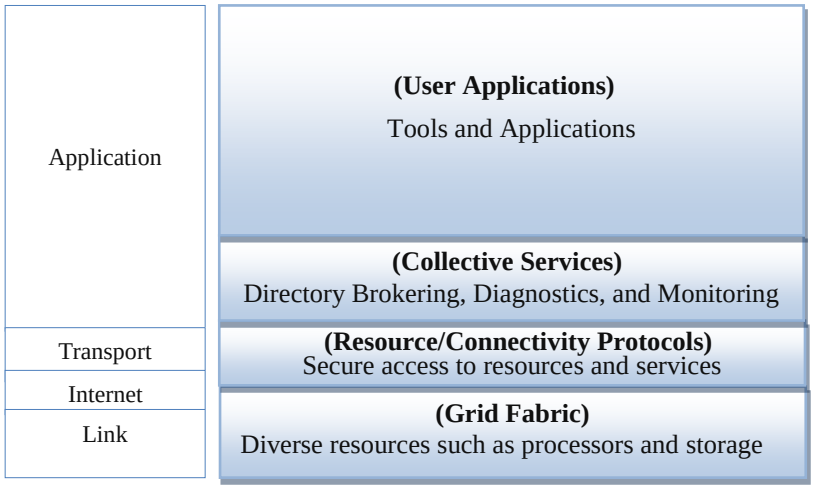

Figure 2a Internet Protocol

Figure 2b Grid Layered Architecture

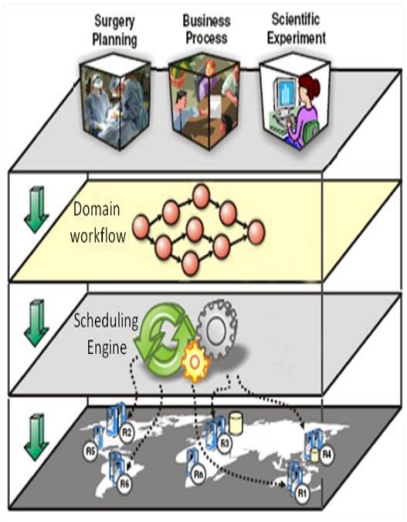

Figure 2c High-level Grid Process of execution

Figure 2: Grid Layered Architecture in relationship to the Grid Process Execution.

In this work, we focus on the Grid/Cloud scheduling component located in the collective service layer. Scheduling in the Grid/Cloud is the process that executes inter-dependent tasks on capable distributed resources at specific times. In addition to the allocation of tasks to capable resources at specific times, the scheduling problem in the Grid/Cloud requires the allocation to satisfy the tasks, as well as the resources objective functions. Figure 2c depicts the high-level Grid/Cloud process of executing tasks and mapping this process to the Grid/Cloud layered architecture. At the top level of the figure, there are different domains that have specific tasks to be executed. Those tasks are modeled through the workflow application with specific QoS that is required to achieve. Modeling the workflow of the domain belongs in the application layer of the Grid/Cloud. This generated workflow is pushed to the scheduling engine for processing. This scheduling engine belongs in the collective service layer of the Grid/Cloud. The scheduling engine considers the different resources in the environment for executing the tasks. Those resources also have specific quality measures to be achieved when processing tasks. The scheduling engine connects to those resources through the connectivity protocols shown

in Figure 2b that enable the connectivity to those resources that live in the Grid/Cloud fabric layer.

3.2 The Grid/Cloud Scheduling Phases

This work we focus on the Grid/Cloud scheduling. In this section, we introduce the Grid/Cloud scheduling logical architecture as shown in Figure 3. The Grid/Cloud scheduler (GS) receives tasks from Grid/Cloud users, selects feasible resources for these tasks according to acquired information from the Grid/Cloud Information Service module, and finally generates tasks-to-resource mappings, based on certain objective functions and predicted resource performance. Unlike traditional parallel and distributed systems, the Grid/Cloud scheduler does not control Grid/Cloud resources directly, but works as a broker [Berman et al., 2003].

Several challenges are presented while scheduling tasks with QoS and constraints in Grid/Cloud computing. A Grid/Cloud environment consists of a large number of resources owned by different organizations or providers with varying functionalities and able to guarantee differing QoS levels. Therefore, multiple criteria must be considered to optimize the execution performance measure. A scheduler cannot always assign tasks onto resources with the highest QoS levels. Instead, it may use cheaper resources with lower QoS that are sufficient enough to meet the requirements of the tasks. Moreover, completing the execution with a required QoS not only depends on the Grid/Cloud scheduling decision of the scheduler, but also depends on the local resource allocation model of each execution site.

A Local Resource Manager (LRM) is mainly responsible for two tasks: local scheduling inside a resource domain, where not only tasks from exterior Grid/Cloud users, but also tasks from the domain's local users are executed, and reporting resource information to Grid Information Service (GIS). Within a domain, one or multiple local schedulers run with locally specified resource management policies. Examples of such local schedulers include OpenPBS [Openpbs, 2012] and Condor [Condor, 2012]. The Local Resource Manager also collects local resource information by using tools such as Network Weather

Service [Wolski, 1999], and Ganglia [Sacerdoti et al., 2003], and reports the resource status information to GIS.

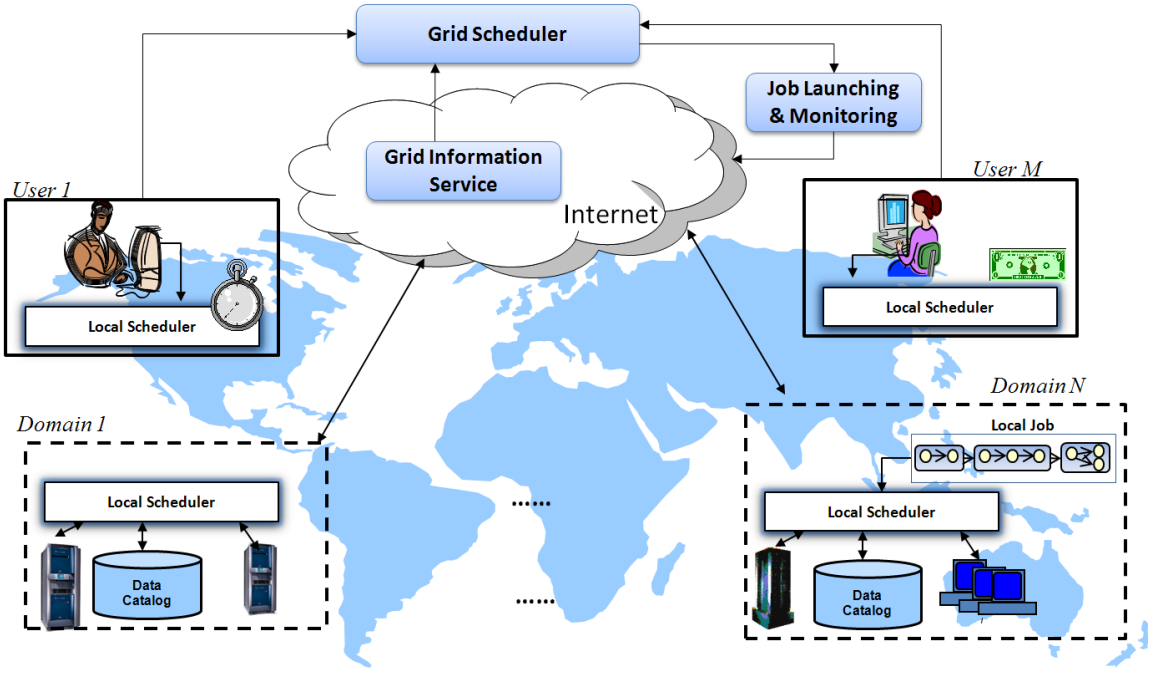

Figure 3: Logical Grid Scheduling Architecture.

Moreover, [Zhu, 2003] proposed a common Grid scheduling architecture. Grid/Cloud scheduling involves three main phases: resource discovery, which generates a list of potential resources; information gathering about those resources and selection of a best set; and task execution, which includes file staging and cleanup. These phases, and the steps that make them up, are shown in Figure 4.

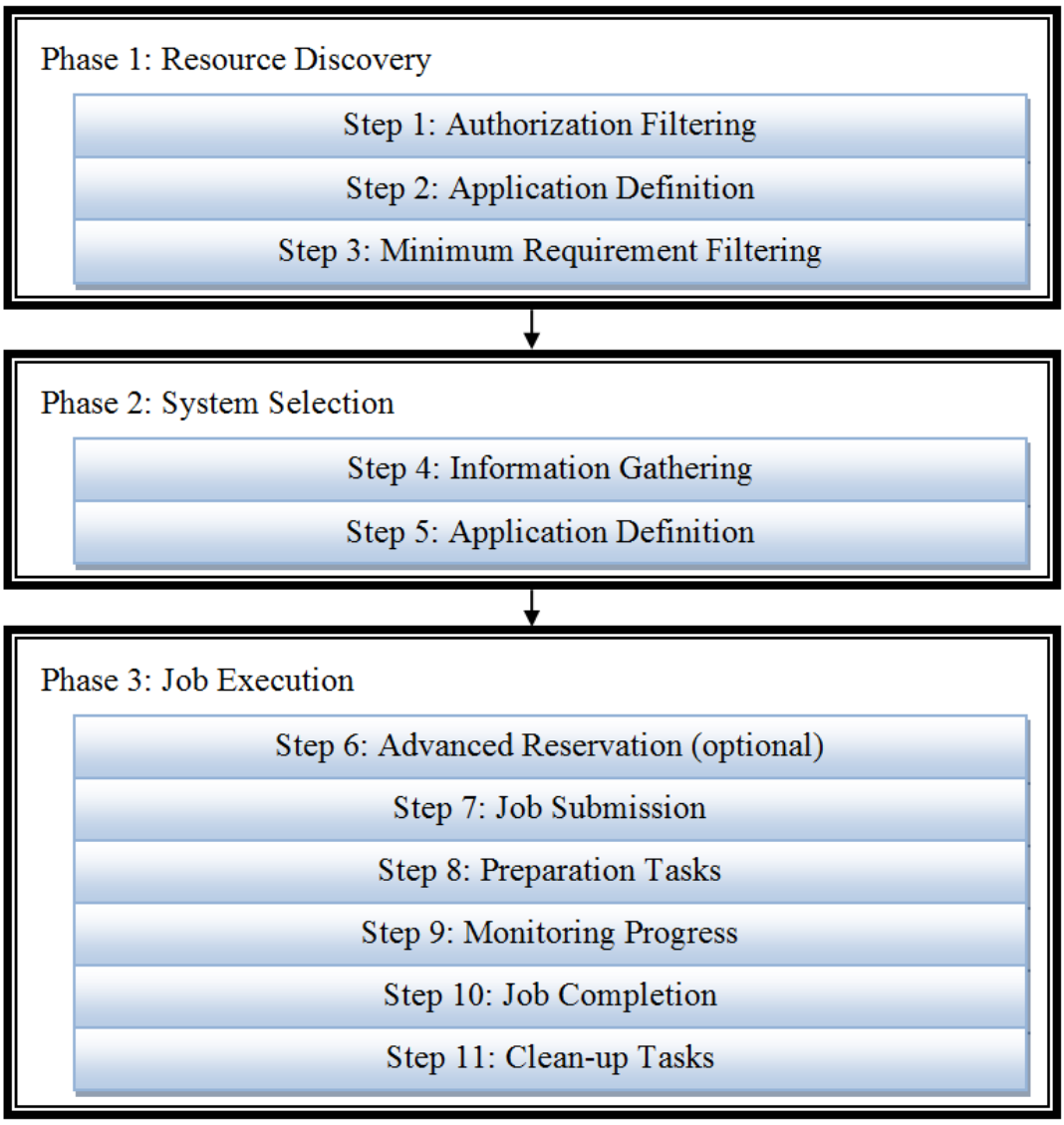

Figure 4: Grid Scheduling Phases.

3.2.1 Phase 1: Resource Discovery

The first stage in any scheduling interaction involves the discovery of the available resources. This involves selecting a set of resources to be considered in Phase 2.

The potential resource selected is the set that has the minimum feasibility requirements. The resource discovery phase is done in three steps: authorization filtering, task requirement definition, and filtering to meet the minimal task requirements.

1) **Authorization Filtering:** The initial step of resource discovery for Grid/Cloud scheduling is to determine the set of resources that exist. At the end of this step the user will have a list of resources to access.
2) **Application Requirement Definition:** The user is to be able to specify the minimum task requirements in order to further filter the set of feasible resources. The set of possible task requirements can include static details such as the operating system or hardware, or the specific architecture as well as dynamic details such as a minimum RAM requirement, connectivity, or space.
3) **Minimal Requirement Filtering:** Given a set of resources to which a user has access and a set of task requirements, the third step in the resource discovery phase is to filter out the resources that do not meet the minimum task requirements. At the end of this step, the user acting as a Grid/Cloud scheduler will have a reduced set of resources to explore.

3.2.2 Phase 2: System Selection

Given the possible resources, all of which meet the minimum requirements for the task, resources must be selected on which to schedule the task. This selection is generally done in two steps: gathering knowledge and making a decision.

4) **Dynamic Information Gathering:** Information about the status of available resources is very important for a Grid/Cloud scheduler to make a proper schedule given the heterogeneous and dynamic nature of the Grid/Cloud computing environment. The role of the Grid/Cloud information service (GIS) is to provide such information to Grid/Cloud schedulers. GIS is responsible for collecting and predicting the resource state information, such as CPU capacities, memory size, service availabilities, network bandwidth, and load of a site in a particular period. GIS can answer queries for resource information or push information to subscribers. An example of a GIS is the Globus Monitoring and Discovery System (MDS) [Czajkowski et al., 2001].
5) **System Selection:** utilizes the gathered information and decides on which resources to use.

3.2.3 Phase 3: Task Execution

The third phase of Grid/Cloud scheduling is running a task.

6) **Advance Reservation (Optional):** Depending on the resource, an advance reservation may be done through some mechanisms or human means.
7) **Task Submission:** Once resources are chosen, the application can be submitted to the resources.
8) **Preparation Tasks:** The preparation stage may involve setup, staging, claiming a reservation, or other actions needed to prepare the resource to run the application.
9) **Monitoring Progress:** Depending on the service and its running time, users may monitor the progress of their services.
10) **Task Completion:** When the task is finished, the user needs to be notified.
11) **Cleanup Tasks:** After a task is run, the user may need to retrieve files from that resource in order to analyze the data. Any of the current systems that do staging (Step 8) also handle cleanup. Users generally do this manually after a task is run, or by including clean-up information in their task submission.

3.3 Characteristics of the Grid/Cloud System

There are two major entities in the Grid/Cloud environment: consumers (requesters) who submit tasks, and providers who share their computation power and services to execute the requests. Those two entities usually have different objectives to be achieved. For example, providers are concerned with the performance of their processors, such as processor utilization, and the consumers are concerned with having their tasks completed as soon as possible.

We explore different definitions of the Grid/Cloud environment and extract the Grid/Cloud characteristics from each definition as presented in Table 1.

Table 1 Grid/Cloud Definitions and Characteristics

Definition	Characteristic
"A type of parallel and distributed system that enables the sharing, exchange, selection, and aggregation of geographically distributed "autonomous" resources depending on their availability, capability, cost, and user QoS requirements". [Buyya, 2002]	Resource Sharing, Autonomy, Scalability, Dynamic, QoS
"Cloud Computing is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on a service-level agreement". [Buyya et. al., 2008].	Autonomy, Scalability, Dynamic, Resource Sharing, QoS
"Computational grids are large-scale high-performance distributed computing environments that provide dependable, consistent, and pervasive access to high-end computational resources" [Foster and Kesselman, 1998]	Scalability
"The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" [Foster et al., 2001]	Resource Sharing, Autonomy
"A distributed network computing (NC) system is a virtual computer formed by a networked set of heterogeneous machines that agree to share their local resources with each other. A Grid is a very large scale, generalized distributed NC system that can scale to Internet-size environments with machines distributed across multiple organizations and administrative domains" [Krauter et. al., 2002]	Heterogeneity, Resource Sharing, Reliable, Scalability, Autonomy
"Grid technologies and infrastructure support the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations - that is, the creation, from geographically distributed components operated by distinct organizations with differing policies, of virtual computing systems that are sufficiently integrated to deliver the desired QoS" [Foster et al. 2002]	Resource Sharing, Dynamic Decentralized, Autonomy, QoS
"A Grid is a system that coordinates resources that are not subject to a centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service" [Grimshaw, 2002]	Resource Sharing, Decentralized, Heterogeneity, QoS, Autonomy
"Cloud Computing, in which not just our data but even our software resides within the Cloud, and we access everything not only through our PCs but also Cloud-friendly devices, such as smart phones, PDAs... the megacomputer enabled by virtualization and software as a service." [McFedries, 2008]	Heterogeneity, multi-tendency, Resource Sharing, Autonomy

In this work we focus on the scheduling problem in the Grid/Cloud environment. The scheduling problem is the allocation of tasks to capable processors at specific time and satisfies specific criterion. From Table 1, we present the characteristics of the Grid/Cloud environment in the context of the scheduling problem.

- **Autonomy**: Grid/Cloud entities are autonomous which means decisions cannot be imposed upon entities. Hence, the scheduler control is distributed among different entities which means that the scheduling solution is distributed among different entities.
- **Heterogeneity**: a Grid/Cloud involves a multiplicity of entities that are heterogeneous in nature. Grid/Cloud nodes both software and hardware can vary. In the context of scheduling, approaches and techniques can be different. For example, an entity in the Grid/Cloud can derive its scheduling solution using the revised simplex optimization technique, where other entity can use other heuristic techniques such as the min-max search to find a solution.
- **Dynamic**: in a Grid/Cloud, entities availability can change at any given time. This means that the scheduling problem model is changing. The number *m* for processors (provider entities), and *n* of tasks (requests) are often changing which means the objective and the constraints in the model are also changing.
- **Resource sharing:** entities have capabilities and power that are shared with other entities in the Grid/Cloud. This means that each entity has the will to provide knowledge as well as sharing the capabilities with other entities.
- **Decentralized:** the knowledge and control of the entities in the Grid/Cloud are distributed. This means that entities hold parts of the scheduling problem model. Moreover, the control is also distributed where parts of the scheduling solution is derived by different entities.
- **QoS:** a Grid/Cloud must assure the delivery of services under established Quality of Service (QoS) requirements. This means that the tasks (requesters), as well as processors (service providers) have quality measures that can be different and sometimes conflicting.
- **Scalability**: the Grid/Cloud has no predetermined number of providers and requesters. This requires the scheduler to scale for a large number of providers and requesters.

- **Reliability**: the Grid/Cloud must be reliable when confronted with requests. This means processing requests must function without failure under given conditions, such as some processors not being available during a given time period.

Chapter 4

4 Scheduling Problem in the Grid/Cloud

The Grid/Cloud as described in the previous sections is a decentralized environment. This means that the scheduling decision making is distributed among entities in the environment. In other words, the knowledge of the scheduling problem and the control are distributed. The Grid/Cloud has two main types of entities: Providers and Consumers. This section introduces the categories of entities in the Grid/Cloud and their characteristics, a static view of the Grid/Cloud model, and the approach of modeling the Grid/Cloud scheduling problem as an economic-based model.

4.1 Overview

We view the representation of requests within the Grid/Cloud using workflows, where tasks are linked according to service dependencies, data flow and computation dependencies. We can classify a workflow as computation intensive when the computational requirements for tasks are high. Similarly, we can classify a workflow as data intensive when data requirements such as storage space or data size are high.

Scheduling a workflow is a process of finding the mapping of tasks in a workflow to the suitable resources so that the execution can be completed with the satisfaction of objective functions, such as execution time minimization. Existing workflow scheduling approaches are non-coordinated, where workflow schedulers perform scheduling related activities independent of the other schedulers in the system. They directly submit their tasks to the underlying Grid/Cloud resources without taking into account the current load, priorities, and utilization. This leads to over-utilization or a bottleneck on some valuable resources, while leaving others largely under-utilized. Further, brokering approaches do not have a coordination mechanism. This worsens the load sharing and utilization problems of Grid/Cloud resources. Cooperative decision making for scheduling in an open environment enables an optimized workflow execution considering the dynamic resource behavior in the Grid/Cloud.

Workflows are executed using distributed resources, where services, computation, and data required by the workflow can be retrieved from several hosts where this is possibility of existing multiple hosts that provide similar services or there exist replicas of data files and multiple sources for computational power. Looking at data for example, it has to be staged to a compute resource before any task associated with the data can be executed at the resource. During or at the end of execution of a task, output data is produced. Such data are to be stored for subsequent tasks requiring them. The sites where the output data are stored could be potential sources of data depending on the policy of retaining or deleting the output data.

The computation requirements of these tasks cannot be totally ignored. After the set of candidate data-hosts are found, the tasks have to be assigned to compute-hosts for execution. The mapping of the tasks to compute hosts depends on the objective function.

Scheduling of the tasks in the workflow primarily focuses on some of the objective functions or combination of them: workflow completion time, and maximize the resource utilization.

4.2 Grid/Cloud Providers

In the Grid/Cloud, we have a set of m provider sites denoted by $P = \{P_1, \dots, P_m\}$. Each provider site P_i $(1 \leq i \leq m)$ is contributing their resources to the Grid/Cloud. A resource is a physical device where tasks are scheduled and processed.

Each site has its resource description, which contains definition of the resource that the provider is willing to contribute.

- Computational resource $r_{\mathrm{i}} = (q_{\mathrm{i}}, a_{\mathrm{i}}, e_{\mathrm{i}}, o_{\mathrm{i}}, \mu_{\mathrm{i}})$ which includes the number of processors q_{i}, processor architecture a_{i} such as the dual core, processor speed e_{i}, installed operating system type o_{i}, and available memory μ_{i}.
- Data resources $d_{\mathrm{i}} = (e_i^d, y_i^d)$: contain information about the storage speed e_i^d, and capacity y_i^d.
- Services s_{i}: includes capabilities related to services that a provider site can deliver. We denote the service capabilities as cap_{i} where $s_{\mathrm{i}} = (cap_{\mathrm{i}})$. The capability set can

be presented formally as $cap_i = \{c_i^1, \dots, c_i^{b_i}\}$ where c_i^k is service k in cap_i, $1 \leq k \leq b_i$ and b_i is the number of services belonging to cap_i.

4.3 Grid/Cloud Consumers

Grid/Cloud consumers have requests to be processed by the Grid/Cloud providers. Consumers visualize requests in the form of workflow. A workflow is represented by a directed acyclic graph $G = (V, E)$, where $V = \{t_1, t_2, \dots, t_n\}$ represent the vertices and E represents edges of the graph. Each vertex represents a task t and there are n tasks in the workflow. The edges maintain execution precedence constraints. Having a directed edge from t_x to t_y mean that t_y cannot start to execute until t_x is completed. The elements within the edges can be described as follows:

- A set of tasks $T = \{t_1, t_2, \dots, t_n\}$
- Computational resources $R = \{r_1, r_2, \dots, r_m\}$
- Services $S = \{s_1, s_2, \dots, s_m\}$
- Data $D = \{d_1, d_2, \dots, d_m\}$

The workflow defines a collection of required requests to be fulfilled by the Grid/Cloud such as specific service invocation, or computation requirement to be performed at specific time in a specific order.

The workflow definition includes the following attributes that define the requirements for executing the tasks on the Grid/Cloud environment.

- Processor speed (e_j^R): the required processor clock speed to process the task.
- Processor architecture requirement (a_j): the required processor architecture such as: a 64-bit AMD processor, a 64-bit Intel processor, a 32-bit Intel processor
- Number of processors (q_j): the required number of processors to execute the task.
- Operating system (o_j): the required operating system to execute the task.
- Memory size (μ_j): the required memory capacity.
- Task set (T_j): set of tasks or services (capabilities) that are required to be executed.
- Deadline ($deadline_j$): the time that the task to be completed.

- Setup time (z_j) may be used to designate the time required for retrieving (copying) input data or the time for linking to a needed library. This may be dependent on the sequence of tasks. In this case, $z_{jj'}$ denotes the setup time needed to compute task *j'* after task *j* where $j' \neq j$.
- Start time ($stime_j$): is a time when the task starts processing.
- Ready time (σ_j): the time at which task is ready to be processed.
- Storage capacity(y_j): the required storage capacity for specific task.
- Storage speed (e_j^D): the required storage speed by the task.

4.4 Formulation

The mathematical model is to include the mentioned characteristics from the consumers and the characteristics of the providers to model the completion time of the workflow, and to maximize the resource utilization. The completion time of the workflow objective deals with minimizing the total time taken for the completion of all the tasks in the workflow. This depends on both the communication time involved in staging the input and output files and the computation time to execute them.

4.4.1 Completion Time Formulation

Formulation Notations:

- P_i – Provider *i.*
- R_{ki} – Resource *k* that belong to provider *i.*
- $x_{j,l,k,i}$ – decision variable where its value is either 0 or 1. $x_{j,l,k,i} = 1$ if task *j* that belongs to workflow l is processed on resource *k* that belongs to provider i.
- $I_{k,i}$ – Idle time of Resource *k* that belongs to provider *i.*
- $c_{j,l,k,i}$ – The completion time of the task *j* in workflow *l* on resource *k* that belongs to Provider *i.*
- $c_{j^*,k,i}$ – The completion time of the last task j^* on resource *k* that belongs to Provider *i.*
- $etime_{j,l,k,i}$ – Execution time of task *j* in workflow *l* on resource *k* that belongs to provider *i.*
- $\sigma_{j,l}$ – Ready time requirement for task *j* in workflow *l.*

- $deadline_{j,l}$ – Deadline requirement of task *j* in workflow *l.*
- $stime_{j,l,k,i}$– start time of task *j* in workflow *l* on resource *k* that belongs to provider *i.*
- i' – is a provider where $i = i', or\ i \neq i'$.
- $\gamma_{j,l}$ – storage capacity required by task *j.*
- $\gamma_{k,i}$ – provided storage capacity by resource *k* on provider *i.*
- $q_{k,i}$ – provided number of processors by resource *k* on provider *i.*
- $q_{j,l}$ – required number of processors by task *j* in workflow *l.*
- $e_{R_{k,i}}$ – provided computation speed by resource *k* on provider *i.*
- $e_{R_{j,l}}$ – required computation speed by task *j* in workflow *l.*
- $\mu_{k,i}$ – provided memory by resource *k* on provider *i.*
- $\mu_{j,l}$ – required memory capacity by task *j* in workflow *l.*
- $e_{d_{k,i}}$ – provided data fetching speed by resource *k* in provider *i.*
- $e_{d_{j,l}}$ – required data fetching speed by task *j* in workflow *l.*
- $\varepsilon_{j,k,i}$ – execution time of task *j* in resource *k* that belongs to provider *i.*

Model **1** focuses on the consumers' objective related to minimizing the completion time of the workflow.

$$min\left\{\sum_{j=1}^{n} x_{j,l,k,i} c_{j,l,k,i}\right\} \qquad \forall l,k,i \qquad (1)$$

s.t.

$$c_{j,l,k,i} = \left(stime_{j,l,k,i} + etime_{j,l,k,i}\right) - \left(stime_{j+1,l,k,i'}\right) \qquad (1.1)$$

$$stime_{j,l,k,i} + etime_{j,l,k,i} \leq stime_{j+1,l,k,i'} \qquad \forall j \ and \ i' \geq i \qquad (1.2)$$

$$stime_{j,l,k,i} + etime_{j,l,k,i} \leq deadline_{j,l} \qquad \forall j \qquad (1.3)$$

$$stime_{l,i} \geq \sigma_j \qquad (1.4)$$

$$\sum_{j=1}^{n} x_{j,l,k,i} * \ q_{k,i} \ \geq q_{j,l} \qquad (1.5)$$

$$\sum_{j=1}^{n} x_{j,l,k,i} * \ e_{R_{k,i}} \ \geq e_{R_{j,l}} \qquad (1.6)$$

$$\sum_{j=1}^{n} x_{j,l,k,i} * \ \mu_{k,i} \ \geq \mu_{j,l} \qquad (1.7)$$

$$\sum_{j=1}^{n} x_{j,l,k,i} * \ e_{d_{k,i}} \ \geq e_{d_{j,l}} \qquad (1.8)$$

$$\sum_{j=1}^{n} x_{j,l,k,i} * \ \gamma_{k,i} \ \geq \gamma_{j,l} \qquad (1.9)$$

$$\sum_{j=1}^{n} x_{j,l,k,i} = 1 \qquad (1.10)$$

$$x_{j,l,k,i} \in \{0,1\}, j = 1, \dots n, i = 1, \dots m, k = 1, \dots z \qquad (1.11)$$

$$stime_{j,l,k,i} \geq 0, j = 1, \dots n, i = 1, \dots m, k = 1, \dots z \qquad (1.12)$$

$$etime_{j,l,k,i} \geq 0, j = 1, \dots n, i = 1, \dots m, k = 1, \dots z \qquad (1.13)$$

$$\gamma_{j,l} \geq 0, j = 1, \dots n \qquad (1.14)$$

$$\mu_{j,l} \geq 0, j = 1, \dots n \qquad (1.15)$$

$$q_{j,l} \geq 0, j = 1, \dots n \qquad (1.16)$$

$$e_{R_{j,l}} \geq 0, j = 1, \dots n \qquad (1.17)$$

$$e_{d_{j,l}} \geq 0, j = 1, \dots n \qquad (1.18)$$

$$e_{d_{k,i}} \geq 0, i = 1, \dots m, k = 1, \dots z \qquad (1.19)$$

$$e_{R_{k,i}} \geq 0, i = 1, \dots m, k = 1, \dots z \qquad (1.20)$$

$$q_{k,i} \geq 0, i = 1, \dots m, k = 1, \dots z \qquad (1.21)$$

$$\sigma_{j,l} \geq 0, j = 1, \dots n \qquad (1.22)$$

Model 1: Minimizing the completion time of the workflow.

Constraint 1.1 defines the completion time to minimize. It is based on the execution time of the task *j* on the workflow *l* and the wait time to execute the next task within the workflow. The objective is to minimize the execution of the whole workflow. Constraints 1.2 ensure the precedence constraints between tasks within the workflow where the completion time of the parent task *j* happens before the start of the execution of the child task *j+1*. Constraint 1.3 ensures the completion time of the tasks on the workflow is completed before the required deadline. Constraint 1.4 ensures the executions of the tasks are started by the ready time. Constraint 1.5 ensures that the required number of processors is met. Constraint 1.6 ensures that the required processor speed is met. Constraint 1.7 ensures that the required memory size is met. Constraint 1.8 ensures that the required data fetching speed is met. Constraint 1.9 ensures that the required storage capacity is met.

4.4.2 Resource Utilization Formulation

The formulation presented in Model 2 focuses on the providers' resource utilization in the Grid/Cloud by minimizing the idle time.

$$\min \sum_{k=1}^{z} I_{k,i} x_{j,l,k,i}\,, \forall i \qquad (2)$$

s.t.

$$I_{k,i} = c_{j^*,k,i} - \sum_{j=1}^{n} \varepsilon_{j,k,i} \qquad \forall k \in P_i \qquad (2.1)$$

$$\sum_{k=1}^{z} x_{j,l,k,i} = 1 \qquad (2.2)$$

$$x_{j,l,k,i} \in \{0,1\}, i = 1, \dots m, k = 1, \dots z, j = 1, \dots n, l = 1, \dots h$$

Model 2: Resource Utilization Provider's Objective.

Constraint (2.1) finds the idle time of resource *k* in provider *i*. It is based on the completion time of the last task *j** on workflow *l* subtracted by the sum of the execution of all tasks in the workflow. Constraint (2.2) ensures that a task is scheduled only once.

4.5 Grid/Cloud Scheduling Problem and Structure

The proposed optimization model of minimizing the completion time presents a formulation based on the local decision making independent from any collective decision related to the Grid/Cloud environment. This is based on the entity's local knowledge.

The entities (providers and consumers) in the Grid/Cloud environment are autonomous and responsible for their own decision making. In such case, the scheduling problems have an additional characteristic derived from the nature of the environment, i.e. the overall problem knowledge is not common knowledge. This problem is called distribution of knowledge in the sense that no entity in the environment has a global view of the problem. Accordingly we introduce the following definition.

Definition 1: A Distributed Scheduling Problem is characterized by the knowledge of the problem is distributed among entities and no entity has a global view of the problem.

Further, the nature of the entities in the Grid/Cloud being autonomous requires also decision making capabilities. This means that entities are driven by its objectives and no entity has control over it. We refer to this type of scheduling problem where the knowledge and control being distributed as a decentralized scheduling problems.

Definition 2: A Decentralized Scheduling Problem is a Distributed Scheduling Problem consisting of self-interested entities and are autonomous in their decision making.

An essential characteristic of decentralized scheduling problems is the distribution of control meaning that the strategies of entities cannot be controlled by outside parties, such as other entities in the environment. This characteristic derives from the self-interested nature of an entity in the environment. However, it does not make them non-cooperative. In most cases, self-interested entities have to cooperate to achieve their respective objectives, but any cooperation must be self-enforcing and not enforced by binding agreements through third parties.

A decentralized environment is constructed from entities that are able to perform some functions independently and exercise some degree of authority in sharing such

capabilities. Such entities are put to work in the same spatial-time domain to achieve either a common or separate goals. Moreover, the knowledge is distributed among entities. For example, when entities are geographically separated and/or owned by different people or organizations, each of them have partial knowledge about the global problem to be solved. Clearly, in these cases, the scheduling problems have an additional characteristic derived from the decentralized environment, i.e. the overall problem knowledge does not reside in one entity. We call it the distribution of knowledge in the sense that no entity in the environment has a global view of the problem.

4.6 Privacy: a Required Attribute in the Grid/Cloud

Consider the following example in the Grid/Cloud where we have two entities, *A* and *B*. Each entity has its own resources that can be used to achieve a goal based on private objective and knowledge to the specific entity. In some cases, for entity *A* to achieve a specific goal, it needs to use resources from entity *B*. In such setting, entity *B* must coordinate with entity *A* to reach an agreement. In this example, we observe the interdependency between entity *A* and entity *B* to achieve a specific goal. Interdependency is viewed as a goal relevant interrelationship among actions performed by various entities. For instance, interdependency may exist between two or more entities when each has a specific knowledge or data acquisition that can only be achieved through the use of a shared resource. Another interdependency that may exist is when an entity attempts to acquire specific knowledge or data that is beyond its capability, but it can be achieved with the help of another entity. The solution to this interdependency problem is known as coordination [Ghenniwa, 1996]. Coordination between entities is a class of solutions that provide structure and mechanism to the system to deal with the interdependency problem. Structure refers to the entities pattern of communication and decision-making that are related to coordination. Mechanism is a composition of decision points, coordinated control and interaction devices directed to resolve problems associated with interdependencies. Given such environment, it is essential that entities receive privacy protection in order to safely coordinate with each other. In our everyday interactions, we have a conceptual privacy model that evaluates interactions in order to protect our privacy [Dey et. al., 2002]. PII 2.0 introduces the concept of risk for

evaluating the privacy aspect of interactions. There are multiple influential factors in privacy situations. The work in [Samani et. al., 2012] identifies the elements of privacy situations and proposes a risk assessment model for evaluating the risk of interactions of two entities. This includes elements such as trust level, severity of operation on information, negotiated agreement between entities, relevancy of the type of the requested information and the type of the offered service, sensitivity, cost and criticality of information and the information gain of exposing the information to other entities. The proposed risk assessment model considers all these elements and calculates the risk of privacy violation in a specific interaction [Samani et. al., 2012]. Utilizing the risk assessment procedure facilitates quantifying privacy interactions. It can lead to evaluate privacy interactions in terms of Privacy Protection Level (PPL).

4.7 Privacy Protection Level in the Grid/Cloud

Generally, to find a scheduling solution in the Grid/Cloud environment entities require to interact. Through the interaction, information is shared between entities. There are several levels of information: information collection, information processing and information dissemination. The input of the scheduling system are information related to tasks, tasks requirements (such as deadline and storage capacity), and resource specification (such as computational resources processor speed and storage resource capacity). The output is information related to the schedule for executing tasks on resources at specific time.

Within this context, information collection happens when the scheduling system collects information about tasks requirements and resources specifications of providers. Information processing refers to all operations such as matching of the capable resources to tasks. Information dissemination on the other hand occurs when the tasks along with their information (such as requirements related to deadline) are sent to the providers for execution. Such setting implies a series of interactions between entities within the Grid/Cloud where sensitive information is exchanged. Hence, privacy protection of entities within those interactions is essential.

In any interaction, consumers in the Grid/Cloud evaluate the privacy aspect of interactions by calculating the Privacy Protection Level (PPL). The amount of PPL within a consumer interaction (for instance 8 PPLs) shows that the provider has to provide 8 PPL to protect the privacy of the consumer. The provider also evaluates the maximum PPL they can provide for receiving the information of the consumer. In the Grid/Cloud environment, information is used to build up the knowledge about the scheduling problem. Such information can be tasks requirements, resource specification and final schedules. Privacy protection is required within such information being exchange within the scheduling problem. We measure the privacy protection based on the PPL. Computing the privacy protections level (PPL) is beyond the scope of this thesis, however, we touch into the PPL concept.

In this work, we utilize the few notations to formulate the privacy concept that PPL relates to them. Members of sets in privacy context have a "Type" value. It is used to classify different members of a set in a subset and addresses the subset with a unique name. "Type" in our model is a predicate. $Type(x, A)$ checks if the type of x is A. Since, it is a predicate, it returns true or false. Accordingly:

- "." Is a function. "."(A,B) returns all the members that belong to A and their type is B. Formally, "." Can be expressed as $"."(A, B): \{x | x \in A \wedge Type(x, B)\}$. For ease of usage, "." (A, B) can be written as $A.B$ or $A_{\vec{B}}$. This function can be used in multiple levels. Therefore, it is a valid statement to write A.B.C. it returns the subset of A that has the type B and type C.
- "=" (A,B) or A=B is a predicate to check the equality of the values of A and B.
- "≡" (x, A) is a function that shows x is equivalent to A. it can be written as x≡A.
- $\overset{def}{\longrightarrow}$ is used for defining the concepts of the model.

We formally denote the Grid/Cloud environment that includes all the providers and consumers that are interacting to solve the scheduling problem as W. We abstract an entity's model in this environment by defining the tuple $e_i =< G, OP, I >$.

- I is a set of scheduling information. It can be tasks, resource specification, and scheduling outputs. Also, we abstract the information related to the attributes within

the scheduling problem by <Attributes, Value>. For example, the value of the attribute "execution time" can be 30 minutes; <execution time, 30>.

- G is a set of Goals of an entity. *Goal* is defined at the entity level as a state of the feasible solution that defines the quality of satisfaction of an entity. The information that define the goal can be presented by a set of attributes in the from <attribute, value>. This set is called *Desired Attribute Set* (DAS). Goals have preconditions that must be satisfied and hence, if preconditions are not satisfied, the goal is not achievable. For example, if the goal is to minimize the completion time. To accomplish this goal, we need to have a computation resource with 5GHz or higher. The existence of a resource with such specifications is a precondition for this goal. If such resource does not exist, then no solution is found to schedule such task.
- Op is a set of *Operations.* They are functions that receive information as an input and generate new information as output. Operations refer to processing scheduling information. For example, matching task requirements to resource specifications and generating schedules for entities can be the examples of operations. For instance, when task t_i is sent to scheduling system in which includes information about providers P={P_1,…,P_m} and their resource specifications R_i, applying operations such as matching on t_i and R_i in P can identify the potential providers for t_i. Applying operations such as finding the providers that task $t_i$will be assigned to them (e.g winner determination) generates new information about what provider can execute which task.

When applying *OP* (operations) on scheduling information, the generated information is sensitive and requires privacy protection. Within this section we continue with an elaboration of the PPL concept within the Grid/Cloud scheduling.

In evaluating interaction among entities based on privacy, one of the influential factors is *Purpose* of collecting information [Singh and Bawa, 2007][Dey et. al., 2002]. *Purpose* refers to a set of operations that are applied on information. We formally present purpose in (2) in Text Box 1. For example, the goal of the provider is to finish t_i before $deadline_i$. Achieving this goal requires having *operations* for processing the task specification such as identifying the providers that are eligible for executing the task and

to deliver the result within the requested $deadline_i$. In this case, the *purpose* of collecting task specification is to apply matching operations on t_i and R_j of P (as the providers).

Exposure in the context of privacy refers to revealing some information to others. It can occur by willingly sharing the information. Also, it is possible by directly or indirectly observing the information. Observation of information has specific information gain [Bezzi, 2007]. In other words, it is possible to obtain more information by observing information. We formally define *Exposure* in (3) in Text Box 1. For instance, an entity can receive the task specification of another entity directly from the owner of the task. It may also collect this information from a third party entity that has the information of an entity. In all of these cases, task specification is exposed.

Privacy violation prevents an entity from achieving their goals. In this context, the focus is on goals that have no conflicts. For example, assuming there is a provider P_k that is capable of executing task t_i and there are consumers Con_i and Con_j that are competing for R_i in P_k. If Con_i exposes t_i to an entity to utilize its service (such as resource discovery service within the brokering paradigm to discover the potential providers for executing t_i), and the entity (such as the broker) shares t_i with Con_j. It is possible that Con_j generates a task that causes that R_i in P_k be allocated to Con_j and no longer be available for executing t_i . In this scenario, another goal of Con_i which is executing t_i within its deadline is not achievable anymore. Hence, if exposing some information is the precondition of achieving a goal and it causes another goal of the entity not to be achieved, then their privacy is violated. We formally define *privacy violation* in (4) in Text Box 1.

Entities are concerned about their *sensitive information*. When having specific information facilitates *privacy violation* of an entity, then that information is considered as sensitive. We formally present *sensitive information* in (5) in Text Box 1. For instance, because exploiting task information can result in privacy violation of consumers, task specifications are sensitive information. Similarly, information such as task specifications resource specification, capability and result of scheduling are considered as *sensitive information*.

In order to avoid privacy violation, there are several privacy protection techniques. For instance anonymization, signing the contracts that support privacy right. Operations owned by a provider can be facilitated with these techniques. Therefore, *Privacy protection* is applying operations that prohibit privacy violation. This is formally defined in (6) in Text Box 1. However, these operations might not cover all aspects of privacy. As an example, K-anonymity is an anonymization technique that is utilized in publishing data sources [Sweeny, 2002]. This technique concentrates on information dissemination and do not address information collection and processing. Moreover, it can be circumvented, if it is used in environments such as the Grid/Cloud. Therefore, there is a probability that an operation that is equipped with *privacy protection* techniques will prevent *privacy violation*.

Privacy Protection Level is the minimum probability of *privacy protection* in operations of a provider. We formally present PPL in equation 7 in textbox 1. The risk assessment procedure [Samani et. al., 2012] identifies the influencing element of privacy in interactions that ultimately can be utilized and result in evaluating PPL.

In this work, we assume that utilizing PPL as the unit of evaluating privacy in interactions is acceptable by all entities and they use PPL as a standard measure for expressing their privacy preferences.

(1) $W =< E, X > E = \{e_i|n\}$ *where* $n \geq 2$ W *is Grid Environment* e_i *is Entity*, X *is Scheduling Domain*
$X = [< Attribute, Value >]$, $DAS = Desired\ Attribute\ Set \in X$
$e_i =< G, OP, I >. G = [g \in X | DAS \subset g]$,
G is a set of Goals, Op is a set of Operations, and I is a set of Information
$Info = tasks|resource\ specification|bidding\ values|constraints|objectives$

(2) $E.e_i.Purpose \xrightarrow{def} \exists s =< Att, Val > \wedge \exists g \in E.e_i.G \wedge s \subset g \wedge \exists pu \in E.e_i.Purpose$, $pu(E.e_i.I) \rightarrow s \subset current\_state\_of\_the\_world$ *(operation)*

(3) $Exposure < Info, Recipients > \xrightarrow{def} [\forall Rec \in Recipient \mid Rec.I \equiv Rec.\underrightarrow{I_{BeforeExposure}} \cup tasks]$

(4) $Privacy\_Violation < Consumer, Others, Info, Purpose > \xrightarrow{def}$

$$\begin{cases} [\exists other \in Others \wedge \exists g1, g2 \in Consumer.G, g1 \text{ has no conflict with } g2 \mid Info \in Consumer.I \wedge \exists i \in Info, i \text{ is precondition of } g1, \\ \quad Exposure < i, other > \rightarrow DAS \notin g2 \vee g2 \text{ is costly achievable}] & \text{if Purpose is Nil} \\ \exists pu \in Purpose, Privacy\_violation < Consumer, others, pu(Info), nil > = TRUE & \text{if otherwise} \end{cases}$$

(5) $Sensitive\_Information < Consumer, Info > \xrightarrow{def} Info \subset Consumer.I \wedge \exists i \in Info, Privacy\_Violation < Consumer, E - consumer, Info >$

(6) $Privacy\_Protection < Consumer, Others, Info, operation > \equiv \neg Privacy\_Violation < Consumer, Other, op(Info), nil >$

(7) $Privacy\_Protection\_Level < Consumer, Provider, Info >$

$$= Min\left(\begin{matrix} Prob((Privacy\_Protection(Consumer, Provider, Info, op) = TRUE) \\ \left[\forall op \in Provider.OP \mid Info \in \underrightarrow{op_{Domain}}\right] \end{matrix}\right)$$

Text Box 1: Privacy Concepts and Principals.

4.8 Economic-Based Model: a Proposed Model for the Grid/Cloud Scheduling Problem

In this section, we map the scheduling problem model on the Grid/Cloud environment into an economic model. This is because of the decentralization nature of the Grid/Cloud environment. In an economy, decentralization is modeled in the context of self-interested rational agents that attempt to achieve their own goals.

In the Grid/Cloud, there are two types of entities, providers and consumers. A consumer attempts to optimize its individual performance objectives only by obtaining the services it requires. Similarly, providers allocate its services and resources to consumers based on its individual satisfaction to their objective. In an economic model of the Grid/Cloud, the applications or service requests belong to consumers. Resources such as CPU, memory, storage and services provided are owned by providers. Scheduling is to allocate the resources (owned by providers) to tasks (belong to consumers) at a specific time.

Consumers on the Grid/Cloud have requirements for the tasks to be considered for execution such as, deadline and precedence constraints between tasks in a workflow. On the other hand, items or resources owned by providers have reserve values related to the time of when a task is executed, resource specifications and the capacity to be consumed. The reserve value is a real number that presents the preference of provider based on specific measure such as resource utilization. Consumers might prefer some requirements over others. The preferences are represented by utility functions. These functions map the requirements of the consumer to a real number. For example, a consumer can prefer a task to be executed on an earlier time than a later time, or resources with higher capacities.

Let J_r denote the set of task requirements and a value v_{J_r} for requirement set J_r, and v_R denote the reserve value for the providers for some resources in R. We present a feasible schedule sch that satisfies J_r and resource R is capable of achieving the task's requirements and $v_{J_r} \geq v_R$.

Let $U(\cdot)$ present the utility function and let sch_1 and sch_2 be two schedules that satisfy both the consumer and providers objectives and reserve value. An agent (consumer and provider) prefer $sch_1 \succ sch_2$, when $U(sch_1) > U(sch_2)$. An agent has a preference on schedules, such that $sch_1 \succ sch_2$, which implies that schedule sch_1 is preferred to schedule sch_2, where $sch_1, sch_2 \in sch$, and sch is all possible schedules. An agent has an indifferent preferences to the schedules sch_1 and sch_2 if $sch_1 \sim sch_2$. Preferences are transitive, if $sch_1 \succ sch_2$ and $sch_2 \succ sch_3$, implies $sch_1 \succ sch_3$ (where $sch_3 \in sch$).

Each schedule $sch_1, \dots, sch_n \in sch$ contains different allocation of the workflow to resources that belong to providers. We present each allocation of a vertex on the workflow as a partial schedule of sch_i.

To formally present partial schedules for an agent, we define the variable $stime_{j,l,k,i}$ to present the starting time for executing task $j \in J_l$ on resource $R_k \in P_i$ in a schedule.

We can express a schedule by a set of tasks starting time on a resource assignment for each task as:

$$sch = \{stime_{j,l,k,i} | J_l \in J, j \in J_l, k \in R_k, P_i \in P\} \forall j \in J_l \quad \textbf{(Equation 1)}$$

To present each schedule for an agent $g \in N$: $sch^g = \bigcup_{J_j \in J} \bigcup_{k \in J_j} stime_{j,l,k,i}$ and hence we have: $sch = \bigcup_{g \in N} sch^g$ where sch^g is a subset of sch, call it a partial schedule of agent g.

A *partial schedule* can be defined as a subset of schedule sch. If a set $sch^g (sch^g \subset sch, g \in N)$ contains only the starting time for executing a task on a resource for a workflow of agent g, we refer to sch^g as a partial schedule for consumer agent g.

To expand the partial schedule to include the privacy protection level, we expand the definition of *sch* to also include the privacy protection level. Hence, we have:

$$sch = \{stime_{jlki} | J_l \in J, j \in J_l, k \in R_k, P_i \in P, ppl_k \geq ppl_j\} \forall j \in J_l \quad \textbf{(Equation 2)}$$

If all constraints of agent g are satisfied in a partial schedule sch^g, then sch^g is a feasible partial schedule for agent g. The formulation presented by Equation 2 presents the feasibility by also considering the Privacy Protection Level (PPL) value to be achieved by provider entities. The overall feasible schedule to in the Grid/Cloud scheduling problem is the union set of feasible partial schedules.

4.8.1 Mapping to the Combinatorial Allocation Problem

In the Combinatorial Allocation Problem (CAP) there is a set of agents *N* and a set of items *M*, held by each provider. Let n = |N| and m = |M| be the numbers of agents and items respectively. A bundle B is a subset of items I. Let B_g where $B \subseteq I$ be a bundle allocated to agent g. An allocation is feasible if $B_g \cap B_{g'} = \emptyset$ for $g \neq g'$.

Each agent has a valuation function over bundles $v_g: 2^M \rightarrow \mathbb{R}^+$. 2^M denote the set of all subsets of M. The set of all bundles including the empty bundle. Valuations are defined over bundles rather than just items. This permits complements and substitutes. Items are complements when their value together is more than the sum of their individual values, and they are substitutes when the reverse holds.

In the Grid/Cloud, items are the processing times supplied by providers. In CAP, items or goods are discrete, where the concept of time in scheduling is continuous. To map the resource processing time to the set of distinct items, we impose a discretization on time horizon of resources to be scheduled. Assume that all time related parameters in the Grid/Cloud scheduling problem, such as the release times, deadlines and processing times, are of integer value of a basic time unit, denoted by τ. Formally, let $[0, \Omega]$ be a time horizon of the resources being scheduled. For each resource k the time units associated is presented as $\tau_{k,\alpha}$, $1 \leq \alpha \leq \Omega$. The set of all resource specific time units within the time horizon can be seen as the set of items I to be sold in CAP, $I = \{ \tau_{k,\alpha} | R_k \in P_i, 1 \leq \alpha \leq \Omega \}$ where R_k is the a resource that belongs to provider P_i. Furthermore, we can also present the privacy protection level as items in the marketplace as: $I_{ppl} = \{ \varrho_{k,\tau} | R_k \in P_i, 1 \leq \tau \leq \Omega \}$ where $\varrho_{k,\tau}$ presents the privacy level protection level given by resource k at specific time unit τ. Any subset $B \subseteq I$ is called a bundle.

Feasible partial schedules are mapped into the concept of bundles in CAP. Agents in the Grid/Cloud value specific allocations not just for items but bundles of items which signifies the preference among partial schedules. In other words, the value function of an agent defines the values that the agent has over the combinations of items. For agent g, the value function in the CAP v_g is defined as: the value of a bundle B is set to the value of the optimal partial schedule for agent g covered by the bundle B, denoted by sch_*^g. That is $v_g(B) = v_g(sch_*^g)$. If no feasible partial schedule is covered by B, $v_g(B)$ is set to zero.

To find the global scheduling solution in the Grid/Cloud, we need to compute the solution to the social choice function $f: (sch^1, \dots, sch^l) \rightarrow S$, that selects the optimal schedule $S^* = f(sch)$ based on the preferences of all agents. The social choice function selects an outcome to maximize total valuation over agents.

$$f(sch) = \max_{s \in S} \sum_g v_g(sch^g)$$

4.8.2 Combinatorial Auction Model

The approach to the Grid/Cloud scheduling problem is to adopt an integrated solution. Auction theory has been applied to the design of a number of real-world markets. In particular, a considerable body of research has been devoted to designing auctions for combinatorial allocation problems (CAPs). As we have mapped the Grid/Cloud scheduling problem to a class of CAPs, it is natural to think of applying combinatorial auctions to the Grid/Cloud scheduling problem. An auction provides a protocol that allow agents to indicate their interest in one or more resources and that uses these indications of interest to determine both an allocation of resource and a set of payments by the agents. In an auction, we have a set of bidders $N = \{1, \dots, n\}$ and a set of goods $G = \{1, \dots, m\}$. Let $v = (v_1, \dots, v_n)$ denote the valuation functions of the different bidders.

CAPs are decentralized problems which involve the complexities at knowledge distribution and control distribution levels. By modeling a CAP as an auction, the levels of complexities are transformed to computational constraints in combinatorial auction design. Kalagnanam and Parkes reviewed four areas of computational constraints, which restrict the space of feasible combinatorial auction mechanisms, including, strategic complexity, communication complexity, valuation complexity, and winner determination complexity [Kalagnanam and Parkes, 2004].

Modeling the resource allocation needs to consider the problem of allocating (discrete) resources among agents using Auction since it provides a general theoretical framework for resource allocation problem among self-interested agents. The nature of the Grid/Cloud market environment is that we have multiple bidders and multiple providers. Hence we establish a formulation in Model 3 that fulfills such characteristics of the market structure and the Grid/Cloud environment providers and consumers.

Formulation Notation:

- *l* – Item
- *l** -- Last item to be processed in the bundle
- *l'* – Item to be executed after item *l*

- $etime_{j,l,k,i}$ – Execution time of item *l* that belongs to bundle *j* executed on resource *k* owned by provider *i*
- $stime_{j,l,k,i}$ – start time of the item *l* of bundle *j* to be executed on resource *k* owned by provider *i*
- σ_j – Ready time for bundle *j*
- $deadline_j$ – Deadline of bundle *j*
- g_p – Provider Agent
- g_c – Consumer Agent
- $v_{reserve}^{g_p}$ – reserve value provider agent g_p

The formulation provided by Model 3 presents the auction model. The auction objective is to maximize the valuation of both providers and consumers. Constraint 3.1 ensures that the bundle of items satisfies the ready time. Constraint 3.2 ensures that the bundle is executed before the deadline. Constraint 3.3 ensures the bundle satisfies the number of processors requirements. Constraint 3.4 ensures that the bundle satisfies the processing speed requirements. Constraint 3.5 ensures that the bundle satisfies the memory capacity requirements. Constraint 3.6 ensures that the bundle satisfies the required data fetching speed. Constraint 3.7 ensures the data capacity requirement. Constraint 3.8 ensures that a bundle is not assigned more than once. Constraint 3.10 ensures the required precedence constraints. The rest of the constraints ensure that all variables are greater than or equal to 0.

$$\max \sum_{g_c \in N} v^{g_c}(B_j) x(B_j, g) + \sum_{g_p \in N} v^{g_p}_{reserve}(B_j) x_{j,g_p} \quad \textbf{(3)}$$

s.t.

$$\sum_{B \subseteq I} stime(B_j, g) \geq \sigma_j, \ \forall g \in N \quad \textbf{(3.1)}$$

$$\sum_{B \subseteq I} stime(B_j, g) + etime_{j,l^*,g_p} \leq deadline_j \quad \textbf{(3.2)}$$

$$\sum_{B \subseteq I} x(B_j, g) * q_g \geq q_l \quad \textbf{(3.3)}$$

$$\sum_{B \subseteq I} x(B_j, g) * e_{R_g} \geq e_{R_l} \quad \textbf{(3.4)}$$

$$\sum_{B \subseteq I} x(B_j, g) * r_g \geq r_l \quad \textbf{(3.5)}$$

$$\sum_{B \subseteq I} x(B_j, g) * e_{d_g} \geq e_{d_l} \quad \textbf{(3.6)}$$

$$\sum_{B \subseteq I} x(B_j, g) * \gamma_g \geq \gamma_l \quad \textbf{(3.7)}$$

$$\sum_{B \subseteq I} \sum_{g \in N} x(B, g) = 1, \ \forall l \in I \quad \textbf{(3.8)}$$

$$x(B, g) = \{0,1\}, \forall g \in N, B \subseteq I \quad \textbf{(3.9)}$$

$$stime_{j,l,g_p} + etime_{j,l,g_p} \leq stime_{j,l',g_p} \qquad j = 1, \dots n, l \neq l' \quad \textbf{(3.10)}$$

$$stime_{l,g_p} \geq 0, l = 1, \dots n \quad \textbf{(3.11)}$$

$$etime_{l,g_p} \geq 0, l = 1, \dots n \quad \textbf{(3.12)}$$

$$\gamma_l \geq 0, l = 1, \dots n \quad \textbf{(3.13)}$$

$$r_l \geq 0, l = 1, \dots n \quad \textbf{(3.14)}$$

$$q_l \geq 0, l = 1, \dots n \quad \textbf{(3.15)}$$

$$e_{R_l} \geq 0, l = 1, \dots n \quad \textbf{(3.16)}$$

$$e_{d_l} \geq 0, l = 1, \dots n \quad \textbf{(3.17)}$$

$$e_{d_g} \geq 0, g \in N \quad \textbf{(3.18)}$$

$$e_{R_g} \geq 0, g \in N \quad \textbf{(3.19)}$$

$$q_g \geq 0, g \in N \quad \textbf{(3.20)}$$

$$\sigma_j \geq 0 \quad \textbf{(3.21)}$$

Model 3: Auction Model

Model 3 by default includes the items through the bundles in the objective. Since, the privacy protection level (PPL) value is defined as an item, it is automatically included in the objective. We still need however to define the PPL constraint to be satisfied in the model. Hence, the following constraint is required in the Auction model to satisfy the expansion of the model to include the privacy requirement in the Grid/Cloud.

$$\sum_{B \subseteq I} PPL(B_j, g) \geq PPL^{g_p}$$

Chapter 5

5 Requirement Based Bidding Language for Resource Scheduling in the Grid/Cloud

In the Grid/Cloud environment, consumers need to be able to describe their preferences for resources that are sold in the marketplace. Furthermore, provider agents need to set their reserve values to be traded in the market. A bid in an auction is an expression of the bidder's preference for various outcomes. In this chapter, we propose a bidding language that is adequate for the Grid/Cloud participants bid/offer specification that facilitates bid/offer description independent from the market mechanism, yet, governs the winner determination mechanism.

In combinatorial auctions, in addition to single items, bidders are allowed to bid on multiple items simultaneously as bundles. This bidding capability presents a challenge to bidders in terms of expressing their values since goods might not have additive value to the bidder. Instead, goods could either be complements or substitutes. Two items are complements when their combined value is larger than the sum of their independent values. For example, service execution requires processing resources and storage resource. On the other hand, if two goods are substitutes, it means they are each worth more when you have just one instead of two. In the Grid/Cloud this can happen when the required computation and services are geographically far apart.

Therefore in a combinatorial auction, a bidder gives the mechanism information regarding the relationship between items. A bidder expresses which items are complements and substitutes by specifying how their value changes for the different bundles. The most straightforward way for the bidder to specify their valuations would be to tell the mechanism the value for every possible bundle. However, specifying a valuation in a combinatorial auction of m items requires providing a value for each of the possible $2^m - 1$ non-empty bundles of items. This representation challenge raises the need for bidding languages that provide some short-hand for placing bids.

Different choices within existing bidding languages vary in expressiveness and simplicity. A well-chosen bidding language aims to strike a balance of the two goals.

- Expressiveness: the ability to express preferences of the entities. The expressiveness is to be short as well as simple of bidders to express their bids.
- Simplicity: the expressed bids should be computationally easy to handle as well as it should be easily understood.

5.1 Grid/Cloud Scheduling Properties

Based on the attribute of the Grid/Cloud environment, we identify the properties that enable an adequate bidding language design. The adequacy is identified based on the properties and requirements of the Grid/Cloud participants (providers and consumers) and the nature of the scheduling problem.

5.1.1 Time-based Requirements and Availability

Consumers have time-based requirements on executing the tasks. Tasks that are executed after the deadline are not desired and may have no value. Hence, the bidding language is to enable the consumers to express their preferences on the time related constraint on their tasks. For example, a consumer requires “Service X” and two CPUs with dual core processors 1Ghz and 4GB of memory, this is to be executed between 12p.m., and 5p.m. The consumers value the service and its execution within this time range at $50. However, they value the execution of the service at $20 between 5p.m., and 11p.m.

Moreover, providers as well can have their resources available at specific time ranges. Based on provider’s objectives they can value specific time ranges to utilize their resources differently. For example, a computational resource between 12am and 7am is valued at $2/hour, and valued at $10/hour between 8am and 5pm.

The bid (and offer) must be able to specify the time ranges within which resources are required and provided, along with the unit of the duration required/provided (i.e. hours, minutes, etc.).

We formally define the time range of requirement by two variables:

- Start time ($stime_{j,l}$ for tasks and $stime_{k,i}$ for providers): $stime_{j,l}$ denotes the feasible start time of the task *j* for workflow *l*. $stime_{k,i}$ denotes the start time on resource *k* owned by provider *i*.
- End time ($etime_{j,l}$ for tasks and $etime_{k,i}$ for resources): $etime_{j,l}$ denotes the latest time of when the task are to finish execution as a requirement for task *j* on workflow *l*. $etime_{k,i}$ denotes the end time of execution on resource *k* owned by provider *i*.

5.1.2 Support for Requirements

The Grid/Cloud consumers have specific requirements towards services, computational resources, and storage resources. Those requirements are to be satisfied when the request is being executed. Such requirements are: computation based (memory, operating system, speed), and storage based (capacity).

We formally define the resource type $A_{R_{ki}}$ as a set of attributes denoted by:

$$A_{R_{ki}} = \{a_1, \dots, a_\alpha\}.$$

$A_{R_{ki}}$ defines the resources owned by provider i such as computational resources and storage. a_α denotes the attributes for the required resource $A_{R_{ki}}$ such as computation speed, memory requirement, etc. for computational resources. Similarly, we denote the required resources for task *j* in workflow *l* by $A_{R_{j,l}}$.

For example, a service in a workflow requires a storage service with at least 25 GB of space, and a computational resource of dual core with the speed of at least 1 GHz, and at least 4 GB memory. We can formalize this as follow:

$$A_{Computation} = \{dual\ core, 1GHz, 4GB\} \text{ and } A_{Storage} = \{25GB\}$$

The bidding Language is to support consumers to express the preferences based on the requirements, as well as providers to describe their reserve values on the capabilities of the resources.

5.1.3 Support for Allocation constraints

Workflows in the Grid/Cloud require a correct execution sequence for the workflow tasks provided by consumers, i.e., an execution that obeys the constraints that embody the logic of the workflow. Such constraint logic is typically of the form, tasks 1 and 2 must both execute (with a possible variation that task 2 executes after task 1) or if task 1 executes then tasks 2 and 3 must execute as well (with the variations that task 3 and task 2 must come after task 1). We define precedence requirements between tasks as $\varrho(T_j, T_{j-1})$. This variable is defined by 0 or 1. It is assigned 1 if there is a precedence constraint between tasks T_j and T_{j-1} which means T_{j-1} is to be executed after T_j. If no precedence constraint between the tasks exists, 0 is assigned to $\varrho(T_j, T_{j-1})$.

5.1.4 Reserve Value on bundles

Providers in the Grid/Cloud own computational power, software services, and storage resources. In the Grid/Cloud market we denote computational processing time, software services, and the storage resources as goods. The providers value those goods based on their objectives. For example, a provider that operates on the resource utilization objective may set the reserve value of the resources lower when the resources are underutilized, and set the reserve value more when there is high demand on the resources. A reserve value is a number that identifies the minimum acceptable value for the good provided by providers.

Providers in the Grid/Cloud may have multiple goods that can be of the same or different types. Providers can set the reserve values on combinations of resources when consumed as a bundle lower than when consumed separately. For example, the provider's reserve value for the combination of a computational resource with 1 GHz CPU speed, 4gb of memory, and a storage space of 40gb is $40. If sold separately, computational resource's reserve value is $35 and the storage space reserve value is $15 which would cost $50 if consumed separately.

We define the resources attributes as $A_{R_{ki}} = \{a_1, \dots, a_\alpha\}$ where R presents the type of resource (computation, storage, service), k is the resource index and i is the provider

index. For example, $A_{computation_{11}} = \{dualcore, 1Ghz, 3gb, windows\ 7\}$ this denotes a computation resource that belongs to provider 1 with the attributes: dual core, 1Ghz, 3gb of memory, and Windows 7 operating system. The reserve value of a resource can be formally presented as: $v_{reserve}^{g_p} = \left(stime_{k,i}, etime_{k,i}, A_{R_{ki}}\right)$. A reserve value of a bundle of resources can be formally presented as: $v_{reserve}^{g_p} = \Big(\left(stime_{1,i}, etime_{1,i}, A_{R_{1,i}}\right), \dots, \left(stime_{k,i}, etime_{k,i}, A_{R_{k,i}}\right)\Big)$.

5.1.5 Consumer's expressiveness on bundles of items and Resource Composites

Consumers in the Grid/Cloud usually demand a combination of different Grid/Cloud resources $A_{R_{ki}}$ with the specific attributes a as a bundle in order to execute tasks at specific time. For example, to execute service X, the consumer requires dual core 1 GHz of CPU power, 3GB of memory, and 100GB of storage within a specific time window. If the providers cannot have all resources required within the required time window, then the execution of the task is not feasible. Generally, bundling does not require the resources to belong to the same provider or the same computational node. However, the resources can be distributed across different nodes in the Grid/Cloud.

Another type of bundling is that the required resources are to be composed in one node. For example, the required service, computational power, and storage capabilities are to be in one specific provider. We denote composite requirements as $\lambda(R_{k-1,i}, R_{k,i})$. This is a binary variable. It is assigned 1 if the resources required $R_{k-1,i}$ and $R_{k,i}$ are to be allocated to the same provider *i* and 0 otherwise.

5.1.6 Sell, Consume Multiple Identical Units of items

In the Grid/Cloud, it is possible to have identical resource specifications that can be provided where $A_{R_{ki}} = A_{R_{k'i}}$, as well as, identical resource requirements needed by consumers where $A_R = A_{R'}$. For example, a provider that has a set of identical computational resources (processing speed and memory size). Also, a consumer that requires five computational resources with the exact CPU requirements and memory.

We denote the number of identical resources required or provided by the notation q_m. The bidding language is to handle such setting with identical resources for both providers and consumers to enable conciseness and expressiveness to the Grid/Cloud bidding language.

5.1.7 Multiple consumers and multiple goods expressiveness

The Grid/Cloud environment by nature includes multiple consumers that require consuming multiple goods that are provided by multiple providers. A Grid/Cloud middleware provides a global directory enabling multiple service providers to publish their resources and multiple consumers' requests to discover them. A market mechanism is to utilize those services and establishes a market structure that enables multiple buyers that consumes multiple goods owned by multiple providers. A bidding language is required to enable the expressiveness for such market structure (multiple goods and multiple consumers).

5.1.8 Trade of Resources

Entities in the Grid/Cloud environment can play different roles. A computation provider becomes a consumer when it requires specific service consumption from another provider. Also, a service provider may require computational resources to execute its services. Trades between entities enables flexibility for entities to utilize their resources and at the same time acquire the resources required within a specific budget.

A bidding language is to enable the expressiveness of providers and consumers of the resources to trade.

5.2 Related work on Bidding Language

Bidding languages addresses valuation complexity portion in the overall market structure. There is a tradeoff in choosing a bidding language between the ease of agent's representation of its preferences, as well as, ease of mechanism's ability to compute an outcome.

Boutilier and Hoos classify the logical biding languages in the literature into two kinds based on the structures of their atomic propositions, namely L_G and L_B [Boutilier and Hoos, 2001]. L_G languages allow bids that are logical formulae where items are taken as atomic propositions and combined using logical connectives. L_B languages use bundles of items with associated prices as atomic propositions and combines them using logical connectives.

Basic bidding languages include OR and XOR or a combination. OR and XOR are logical connectives which can be used to combine atomic propositions of bidding languages. In the OR bidding language every given bundle has an associated value. Bids can be formed by combining any possible bundles and adding their valuations. This is how it would be done if there were no complements or substitutes. For example: $(S_1)\ OR\ (S_2)$, which states that the agent wants S_1 or S_2 or both, has a linear space representation of this valuation function. In the XOR bidding language, a bid is formed by connecting bundle using XOR. For example: $(S_1)\ XOR\ (S_2)$, which essentially allows an agent to enumerate its value for all possible sets of items. This bidding language is simple to interpret, in fact given a bid b in the XOR language, the auctioneer can compute the value *B(S)* for any bundle in polynomial time [Nisan, 2000]. However, this bidding language is not very expressive. XOR bids for this valuation function are exponential in size (explicitly enumerating the value for all possible bundles) [Parkes, 1999b].

[Nisan, 2000] observes that other combinations, such as XOR-of-OR languages and OR-of-XOR languages, allow compact representations of certain preference structures and make tradeoffs across expressiveness and conciseness. The work also proposes an OR* bidding language, which is expressive enough to be able to represent arbitrary preferences over discrete items, and as compact a representation as both OR-of-XOR and XOR-of-OR representations. The work also examines a variety of bidding languages and their properties. For example, we see there that OR (‘‘additive-or’’) bids, which allow the bidder to make non-exclusive offers on bundles, can capture all, and only, the super-additive valuations. In contrast, XOR (‘‘exclusive-or’’) bids, which allow the bidder to make exclusive offers on bundles, can capture all valuations, though they may require an

exponentially longer expression than the OR bids. However, asking an agent to disclose a full valuation function is often not necessary, because many parts of it might be irrelevant for computing the allocation.

The TBBL approach proposed by [Cavallo et. al. 2005] shares some structural elements with the L_{GB} language but has differences in its semantics. In L_{GB}, the semantics are those of propositional logic, with the same items in an allocation to satisfy a tree in multiple places. This can make L_{GB} more concise in some settings, however the semantics TBBL provides is better expressiveness where the value of a component in a tree can be understood independently from the rest of the tree. L_G, L_B, TBBL languages target combinatorial auctions in general. However, they cannot be applied directly to the Grid/Cloud scheduling problems because they are designed based on an assumption: the goods to be auctioned are discrete items. Nevertheless, in Grid/Cloud scheduling, "goods" are processing times on computational resources and services, which exhibit continuity. To deal with this issue, a common approach adopted in the literature is to restrict the continuity of time by imposing a discretization on the scheduling time windows [Wellman et al, 2001].

Nisan [Nisan, 2000] describes the expressiveness of a language, which is a measure of the size of a message for a particular family of valuation functions, and the simplicity of a language, which is a measure of the complexity involved in interpreting a language and computing values for different outcomes.

The expressiveness of a bidding language, or the compactness of representations that it permits, becomes even more important when one considers the agent's underlying valuation problem. Suppose that an agent must solve an NP-hard constrained optimization problem to compute its value for a set of items, with objective function *G* and constraints *C*. In the XOR representation the agent must solve this problem once for every possible input $S \subseteq G$, i.e. requiring an exponential number of solutions to an NP-hard problem.

5.3 Tree Based Specification Bidding Language

We formally define the bidding structure that includes the specifications of consumers and providers. The specifications define both the consumers and providers feasible bundles (partial schedules). We define the resources attributes as $A_{R_{ki}} = \{a_1, \dots, a_\alpha\}$. We denote the required bundle characteristics for a consumer as: Start time $stime_{j,l}$, end time $etime_{j,l}$, required resource $A_{R_{ki}}$, required resource attributes a_α, number of resources q_m, coupling requirements $\lambda(R_{k-1i'}, R_{ki})$ if resources are required to be coupled within the same provider, and precedence requirements $\varrho(T_j, T_{j-1})$. We formally define the bundle based on the required specifications to execute the task. $B_j = \left(stime_{j,l}, etime_{j,l}, A_{R_{ki}}, q_m, \lambda(R_{k-1i'}, R_{ki}), \varrho\left(T_j, T_{j-1}\right)\right)$. The valuation of the bundle is presented by $v_j^{g_c}(B_j)$.

On the other hand, we define the reserve value of the provider based on the provider's resource capabilities and the time interval availability. We denote the reserve value as: $v_{reserve}^{g_p} = \left(\left(stime_{1,i}, etime_{1,i}, A_{R_{1,i}}\right), \dots, \left(stime_{k,i}, etime_{k,i}, A_{R_{k,i}}\right)\right)$.

In this work, we apply the Tree Based Bidding Language (TBBL) proposed in [Cavallo et. al. 2005]. TBBL enables the market combinatorial exchange requirement. However, the Grid/Cloud scheduling requires an addition to the nodes on the TBBL to include specifications related to consumers and providers. Consumers and providers are able to value and trade items in the market based on specifications.

In the Grid/Cloud bidding language we have a tree $Tree_i$ from bidder *i*. Let $N \in Tree_i$ denote a node in the tree, and let $v_i(N) \in \mathbb{R}$ denote the value specified at node *N*. Let $Leaf(Tree_i) \subseteq Tree_i$ be the subset of nodes representing the leaves of $Tree_i$ and let $child(N) \subseteq Tree_i$ denote the children of node *N*. All nodes except the leaves are labeled with the interval operator $\mathbb{I}_{lb}^{ub}(N)$ that is imposed on a node to be satisfied based on the *ub* (upper bound) and *lb* (lower bound) values. The node that has the operator $\mathbb{I}_{lb}^{ub}(N)$ has two partitions:

- The operator – which is satisfied if at least *lb* of the child nodes are satisfied and at most *ub* of the child nodes are satisfied. The use of logical operators is relevant for the bidding language since it enables the users' preferences elicitation. Specially, for the case of bidding specification for Grid/Cloud resources since Grid/Cloud services usually have complex resource requirements. This concludes, if some node N is not satisfied, then none of its children may be satisfied. Along with the operator, we include the time interval in which the execution of the children nodes is feasible. Given *n* number of children, there are three different interval operator types:
  - $\mathbb{I}_n^n$ – This means that all children must be satisfied.
  - $\mathbb{I}_1^n$ – This means that at least one child node is satisfied and at most all children are satisfied.
  - $\mathbb{I}_1^1$ – This means that at least and at most one child is satisfied.
- Value – this expresses the valuation of executing the leaves that satisfy the operator conditions and the workflow requirements. Both the parents and the leaf nodes can express valuations this is to allow complements and substitutes within the bidding language.

Leaf-nodes contain specification of bidder's requirements. Each leaf-node provides expressiveness to capture either buyers' requirements or sellers' offers. Each leaf *N* has three partitions.

- The first partition includes a label for the item to either buy or sell.
- The second partition includes the requirement of a consumer if the first partition is labeled as "buy" or the items that are to be sold in the market if the first partition is labeled as "sell".
- The third partition includes the value.
  - For a consumer the value is related to the expressed requirements.
  - For a provider, the value is a reserve value of the items.

For example, considering an application that requires more than one type of resource (CPU and storage), and multiple quantities of each type of resource, bids should be able to convey preferences on bundles of resources**.** $\mathbb{I}_1^1$ operator permit the expression of substitute bids. By means of $\mathbb{I}_1^n$ operators bidders indicate their willingness to accept

partial satisfaction whilst $\mathbb{I}_n^n$ operators indicate their requirement for complete satisfaction.

5.4 Bidding Language Expressiveness

In this section we explore different scheduling problem structures in the Grid/Cloud environment from both consumers and providers and demonstrate the expressiveness of the proposed bidding language in the Grid/Cloud.

Preference in quantity and time: as shown in Figure 5 the bidder requires five CPUs with a speed of 1Ghz, Intel dual core CPU architecture, at least 2Gb of memory, and storage resource of at least 100Gb SSD drive for 3 hours between 10a.m. and 5p.m., that is, the bidder is asking for a precise time range.

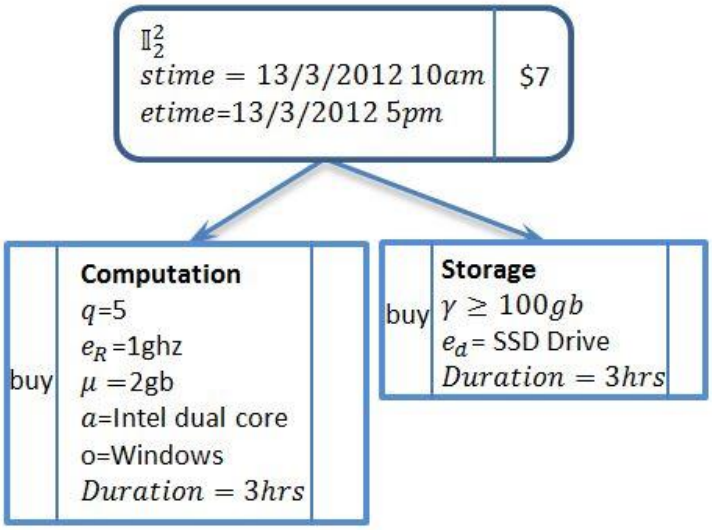

Figure 5: Expressiveness to CPU quantity and time.

Preference in quantity and time to be consecutive: as shown in Figure 6 the bidder requires 5 CPUs with a speed of at least 1ghz, Intel dual core CPU architecture, at least 2gb of memory, and storage resource of at least 100Gb SSD drive for 3 consecutive hours between 10a.m. and 5p.m., that is, the bidder is asking for a precise time duration to utilize the resources within a specific time range. The “consecutive” specification is identified through the pre-emption attribute. This means that once the execution starts,

the CPU processing not to be interrupted. The valuation is placed on the root node to specify the value for the bundle.

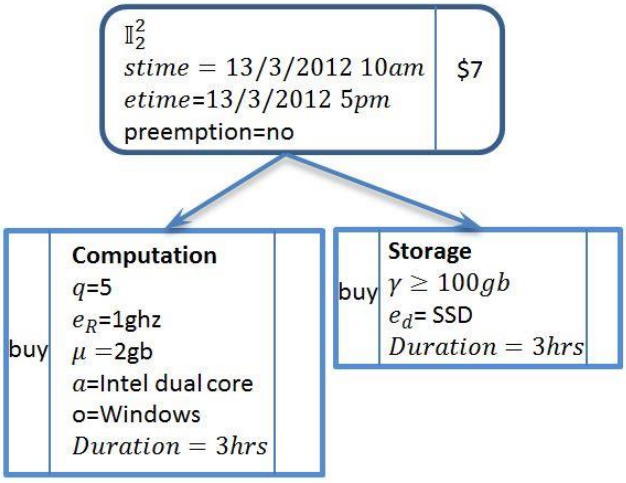

Figure 6: Expressiveness of CPU quantity and time to be consecutive.

Services with precedence constraints that require specific computation and storage: Figure 7 shows the expressiveness of the bidding language for such Grid/Cloud scenario. In the parent node, the *prec* attribute is defined to satisfy the *precedence* attribute defined in the child node.

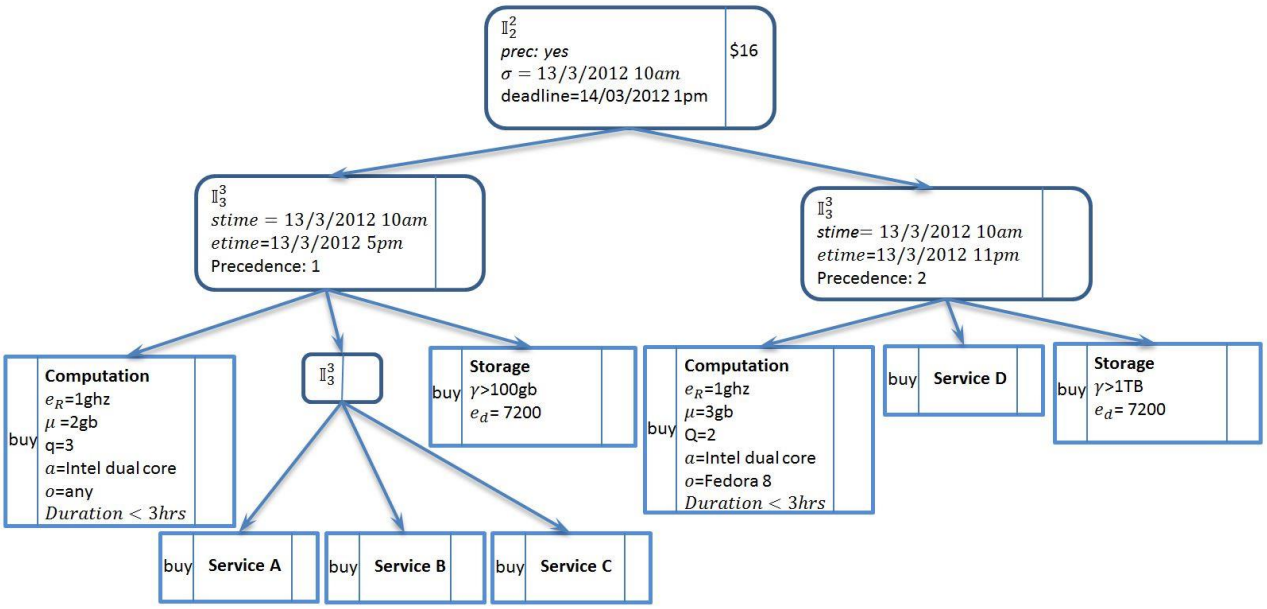

Figure 7: Expressiveness of Service Requirements that has precedence constraints.

Expressiveness to trade resources based on time:

Figure 8 considers two bidders. Bidder 1 potentially sells one of his computation or storage resources that has the specifications shown in Figure 8, if he can get Bidder 2's item (scheduling service) at the right price. Bidder 2 is interested in buying one or both of Bidder 1's items and also in selling his own item. We consider each of the possible trades: If Bidder 1 trades its computation resources for the scheduling service he gets $2 of value and Bidder 2 gets $7. If Bidder 1 trades storage resource for the scheduling service he gets $-2 of value and Bidder 2 gets $2. If no trade occurs, both bidders get $0 value. Therefore the efficient trade is to swap the computational resource for the scheduling service.

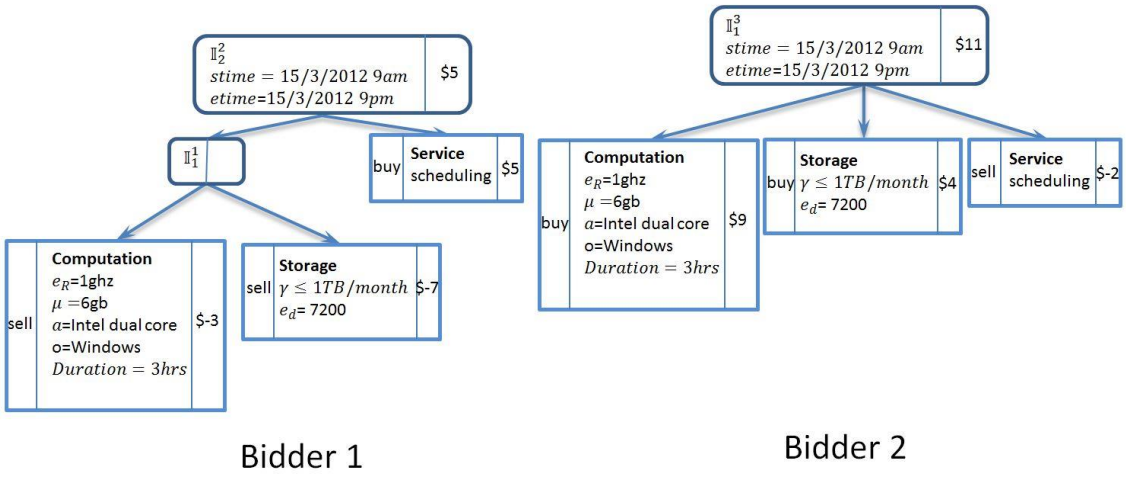

Figure 8: Expressiveness of a Trade Case.

Provider expressiveness to sell services and computation based on time constraints:

Figure 9 shows the provider expressiveness example using the bidding language. The root node expresses that the provider is offering three different bundles that can be consumed. A consumer can purchase all bundles during a specific time window with the reserve value of $38. A consumer can also consume bundles individually since the root node expresses the lower bound to be 1. For instance, a consumer can buy the first bundle that has the reserve value of $7. This bundle includes specific computation and storage specifications expressed on its leaf nodes. Since the lower bound of consuming the leaf nodes is also 1, it is possible to get either the computation or the storage for a reserve

value of $4 for the storage or $6 for the computation. Similar idea is applied to the rest of the nodes on the example.

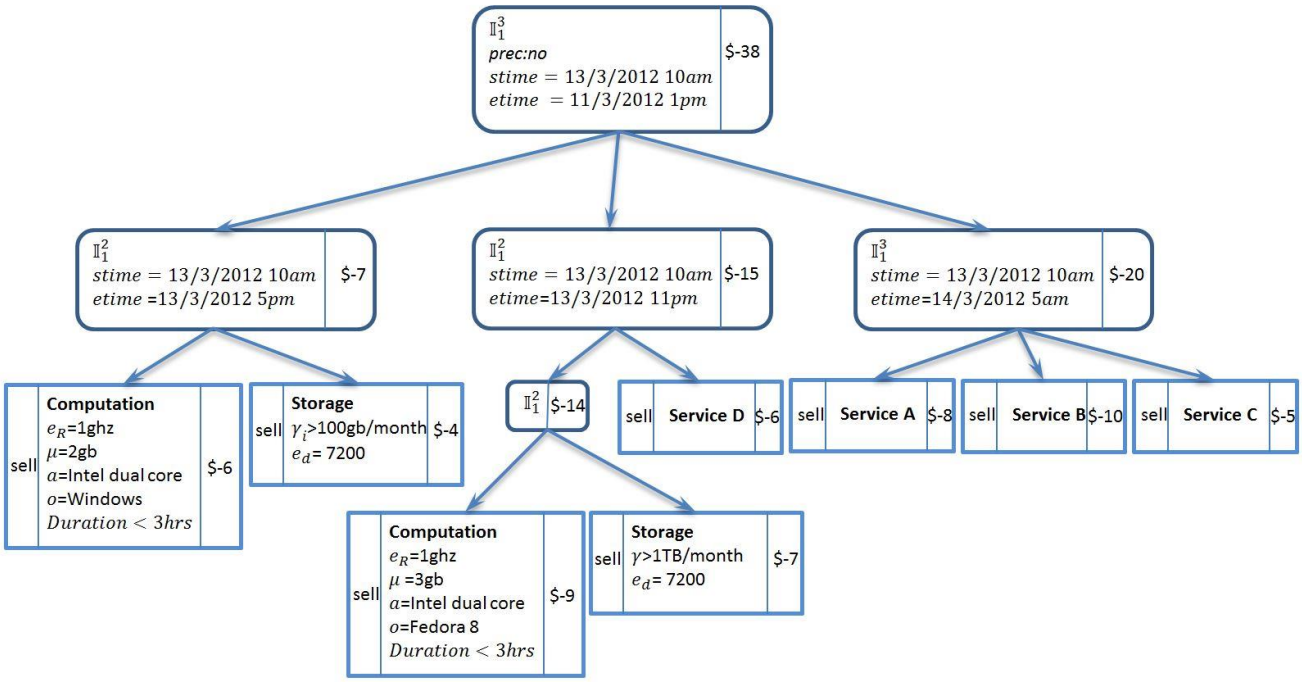

Figure 9: Provider Expressiveness using the bidding language.

5.5 Bidding Language Expansion for the Grid/Cloud Scheduling with Privacy Concerns

In Section 5.1, we discussed the required properties of the bidding language given the nature of the Grid/Cloud environment without considering the privacy attribute. In this section, we expand on the properties to enable the adequacy of the scheduling problem with privacy concerns in the Grid/Cloud environment.

We additionally add the privacy requirement as part of the bidding language properties. Consumers have privacy requirements to execute its services by service providers. Similarly, providers are expected to provide the privacy protection level within the Grid/Cloud environment. Providers within the Grid/Cloud environment are expected to provide at least the required privacy protection level (PPL) as mentioned in the analysis in Chapter 4. Formally the privacy protection level requirement is defined as PPL_g. It is an integer value that describes the privacy level requirement by agent g.

We formally refine the bidding attributes presented in Section 5.3 that includes the specifications of consumers and providers. The specifications define both the consumers and providers feasible bundles (partial schedules). Recall, the definition of the resource attributes as $A_{R_{ki}} = \{a_1, \dots, a_\alpha\}$. We denote the required bundle characteristics for a consumer as: Start time $stime_{j,l}$, end time $etime_{j,l}$, required resource $A_{R_{ki}}$, required resource attributes a_α, number of resources q_m, coupling requirements $\lambda(R_{k-1i'}, R_{ki})$, precedence requirements $\varrho(T_j, T_{j-1})$, and additionally we include the privacy level protection value *PPL*. We formally define the bundle based on the required specifications to execute the task. $B_j = \left(stime_{j,l}, etime_{j,l}, A_{R_{ki}}, q_m, \lambda(R_{k-1i'}, R_{ki}), \varrho\left(T_j, T_{j-1}\right), PPL_j\right)$. The valuation of the bundle is presented by $v_j^{g_c}(B_j)$.

In addition, we refine the reserve value of the provider based on the provider's resource capabilities, the time interval availability, and the privacy protection level value. We denote the reserve value as:

$$v_{reserve}^{g_p} = \begin{pmatrix} \left(stime_{1,i}, etime_{1,i}, PPL_{1,i}, A_{R_{1,i}}\right), \dots, \\ \left(stime_{k,i}, etime_{k,i}, PPL_{k,i}, A_{R_{k,i}}\right) \end{pmatrix}$$

Accordingly we modify the attributes within the proposed bidding language in Section 5.3 to include the consumer's requirement of the privacy protection level (PPL) and the provider's privacy protection level to requests being processed. Figure 10 shows an example of the addition of the PPL attribute to the proposed bidding language. The PPL value is presented in the parent node to reflect on the privacy requirement for the "Buy" leave nodes or the provided PPL for the "Sell" leave nodes.

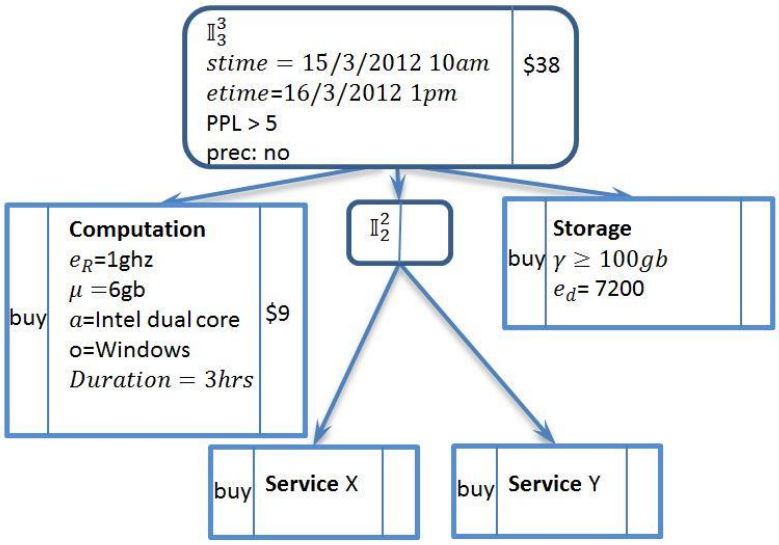

Figure 10: Grid/Cloud Tree Bidding Specification Language with Privacy Attributes Requirement Example.

5.6 Proposed Bidding Language Conciseness

In this section we show the conciseness in comparison with the some of the cases shown in the previous section using the TBBL language. To do this, we first need to divide the time units into slots. We define the time slots that the provider is selling as 1 hour time slots of CPU or Storage. The consumer requires expressing their bidding based on the slots and time requirements. We express the case depicted in Figure 5 using TBBL as shown in Figure 11. Also the case depicted in Figure 6 is expressed using TBBL in Figure 12. We observe that the discretization of the time slots increases the size of the tree and requires expressing all possible time combinations in the feasible space. Figures 11 and 12 show a portion of those combinations. With our approach, we express preferences within a time interval. This cuts down on the size of the tree and makes it more concise.

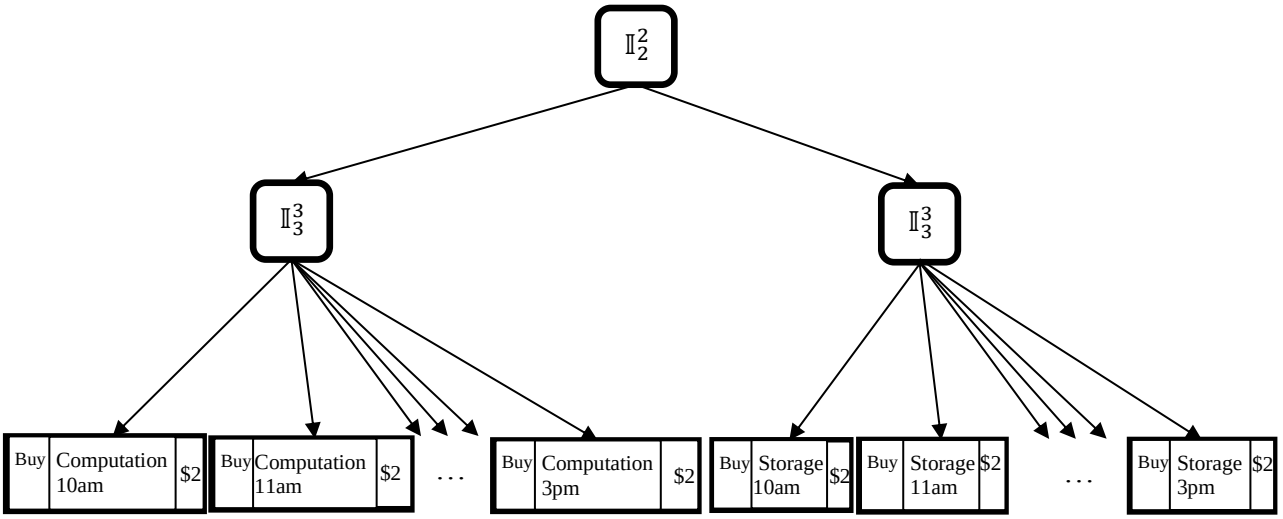

Figure 11: TBBL Representation for the case in Figure 5.

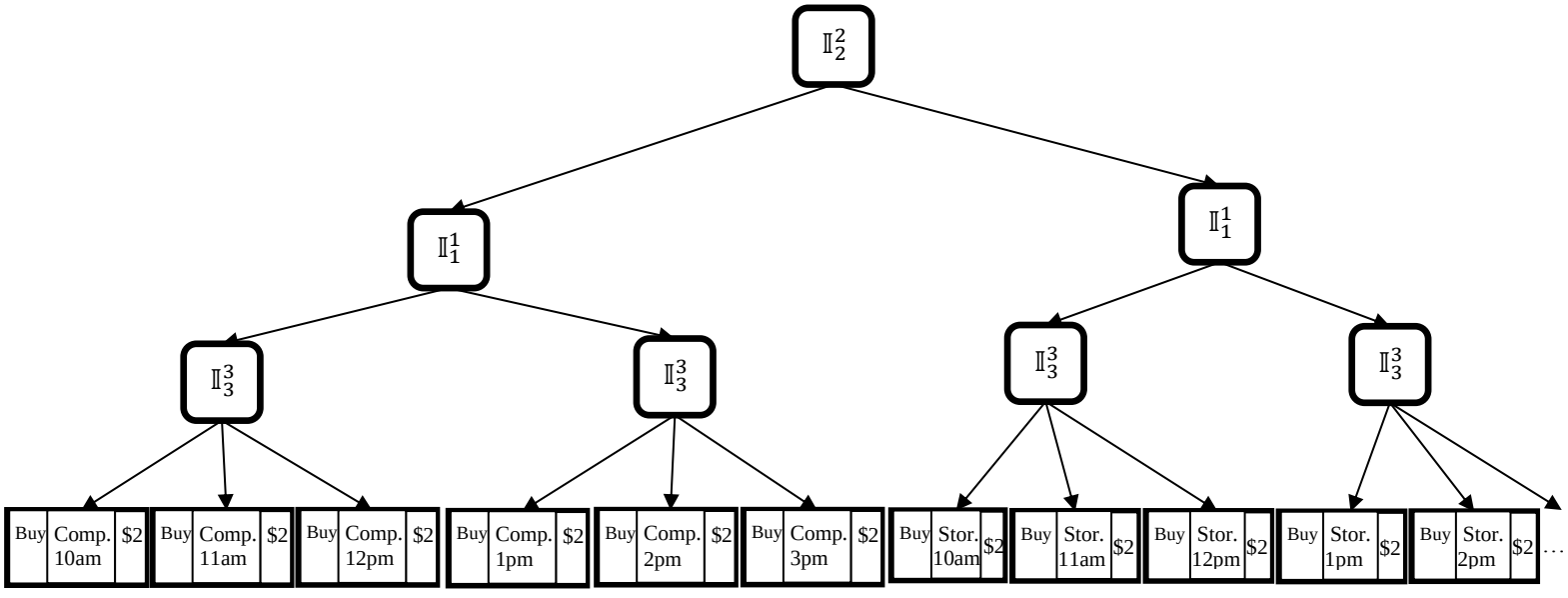

Figure 12: TBBL Representation for the case in Figure 6.

Figure 13 and Figure 14 show the comparison based on the number of nodes (conciseness) between the proposed Grid/Cloud Tree Based Bidding Specification Language (TBBSL) and the Tree Based Bidding Language (TBBL). Figure 13 shows the number of tree nodes increase as the number of item requests increase. Figure 13 focuses on 2 hours continuous time requirement within a time interval of 7 hours. It shows that the TBBL number of nodes highly increases as we increase the number of resources required when comparing with our proposed TBBSL. The reason is that the TBBSL includes in the parent node the properties related to the time requirements which cuts from the representation of the children nodes.

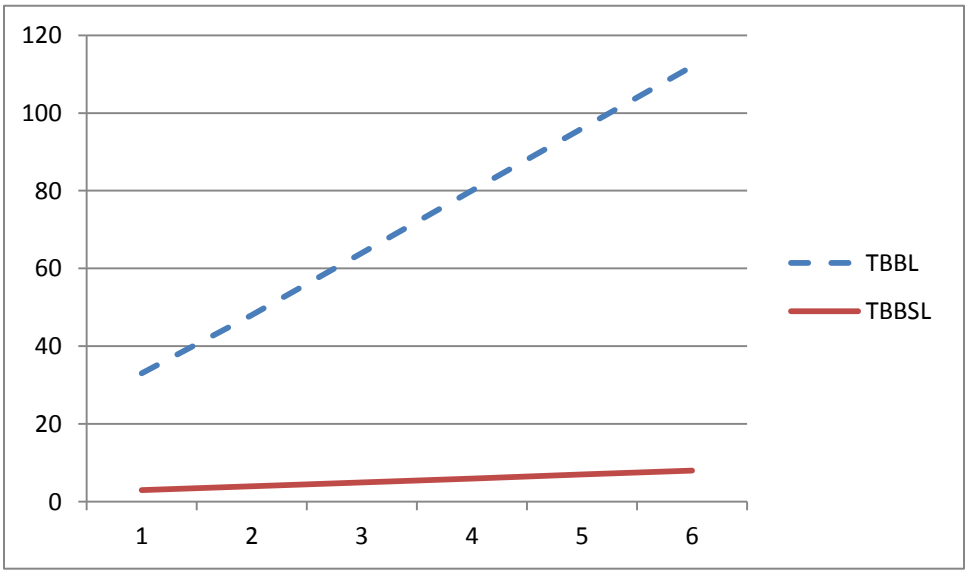

Figure 13: Continuous time requirement for items.

Similarly, we experiment with the representation of the bidding languages with time discontinuous case requirement as shown in Figure 14. In this experimentation, we evaluate based on a time interval of 7 hours and 2 hours of the resources are required without having to be continuous. For example, a resource usage can be from 5 to 6 and again from 10-11. We found that TBBSL is more concise from the TBBL as we increase the number of resource requirement.

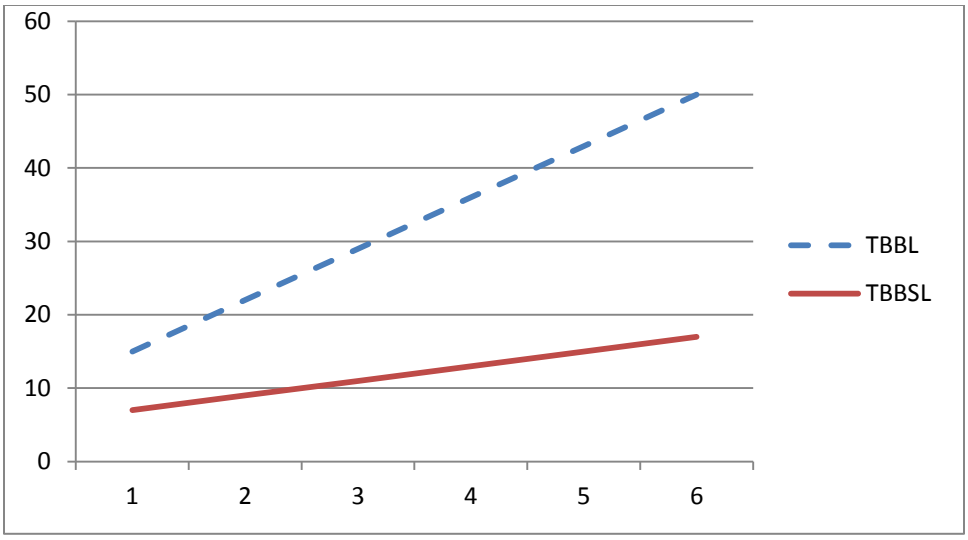

Figure 14: Discontinuous time requirement for items.

Chapter 6

6 Winner Determination

In the Grid/Cloud environment, system designers impose an interaction protocol and independent nodes choose their own strategies which cannot be imposed by an outside entity. Hence, negotiation protocols need to be designed assuming the entities have their own private goals to achieve. In such environment, the aim is on the social outcome given adequate information that enables autonomous entities to achieve optimal resource allocation for the individual and for the society.

Generally, Winner Determination (WD) problem is known as an NP-hard [Rothkopf, Pekec, and Harstad, 1998]. In this chapter, we formulate the WD problem as an Integer program, and propose and adequate mechanism for the Grid/Cloud.

6.1 Market Mechanism Properties

An essential phase in designing a market is to understand the nature of the trading within the environment. The adequacy of the market mechanism for the Grid/Cloud environment is measured based on the following properties:

- Allocative efficiency: An allocation is efficient if the sum of individual utilities is maximized. A mechanism can only attain allocative efficiency if the market participants report their valuation truthfully. This requires incentive compatibility in equilibrium.
- Incentive compatibility: A mechanism is incentive compatible if every participant's expected utility maximizing strategy in equilibrium with every other participant is to report his true preferences.
- Individual rationality: The constraint of individual rationality requires that the utility following participation in the mechanism must be greater than or equal to the previous utility.
- Computational tractability: Computational tractability considers the complexity of computing a mechanism's outcome. With an increasing number of participants,

the allocation problem can become very demanding and may delimit the design of choice and transfer rules.

6.2 The Winner Determination Problem: Formulation

Based upon this bidding language proposed in Chapter 5, the winner determination problem is formulated as an integer program.

Given $Tree_i$, let $N \in \lambda_i$ denote node $N \in Tree_i$ that is satisfied by trade λ_i. We formulate the Requirement-based Tree Bidding Language proposed in Chapter 5 of the WD problem for bid tree $Tree_i = (Tree_1, \dots, Tree_n)$:

$x_i(N)$ – decision variable of selecting node *N* within tree *i*. Its value is either 0 or 1. It is assigned value 1 if selected and 0 otherwise

$v_i(N)$ – valuation of node *N* within tree *i*

$z_t(N)$ – decision variable on allocating node *N* within time *t*

$Q_i(N)$ – quantity of item across Node *N*

$q_{buy}(N, A_R)$ – required quantity of resource specifications A_R of node *N*

$q_{sell}(N, A_R)$ – quantity of resources sold of node *N* with specifications A_R

lb – lower bound of child nodes to be satisfied

ub – upper bound of child nodes to be satisfied

leaf – presents the leaf node

child – presents child node

$$max \sum_i \sum_{N \in Tree_i} v_i(N) x_i(N) \quad \textbf{(4)}$$

s.t.

$$lb\ x_i(N) \leq \sum_i \sum_{N' \in child(N)} x_i(N') \leq ub\ x_i(N),\ \ N \in \{Tree_i \backslash leaf(Tree_i)\} \quad \textbf{(4.1)}$$

$$stime_i\ x_i(N) \leq z_t(N) \leq etime_i\ \ x_i(N),\ \ N \in \{Tree_i \backslash leaf(Tree_i)\} \quad \textbf{(4.2)}$$

$$\sum_{N \in leaf(Tree_i)} Q_i(N)\ x_i(N) \leq \lambda_i \quad \textbf{(4.3)}$$

$$\sum_{N \in leaf(Tree_i)} q_{buy}(N, A_R)\ x_i(N) - \sum_{N \in leaf(Tree_i)} x_i(N)\ \ q_{sell}(N, A_R) \geq 0 \quad \textbf{(4.4)}$$

$$x_i(N) \in \{0,1\},\ \ z_t(N) \in \{0,1\}, \lambda_i \in \mathbb{Z} \quad \textbf{(4.5)}$$

$$Q_i(N) > 0\ , lb > 0, ub > 0 \quad \textbf{(4.6)}$$

Model 4: Winner Determination Problem Formulation.

Constraint (4.1) enforces the interval operator on the parents' nodes. It ensures that no more and no less than the appropriate number of children is satisfied for any node that is satisfied.

Constraint (4.2) enforces the execution of the nodes is within the required time window described in the parent node.

Constraint (4.3) ensures that the quantity of each item across all satisfied leaves is no greater than the total number of units awarded in the trade. This works for providers as well as consumers: for providers a trade is negative, and this requires that the total number of items indicated as sold in the tree to be at least the total number of items traded from the bidder in the trade.

Constraint (4.4) ensures the minimum requirements of the consumers are achieved.

6.3 The Winner Determination Problem: Formulation with Privacy Concerns

Based on the revised property within the bidding language to include the privacy protection level (PPL) mentioned in Section 5.5 and the mapping of the privacy attribute

in Section 4.8 to the economic-based modeling, we expand the winner determination model as shown in Model 5.

$PPL_{buy}(N)$ – denotes the amount of privacy protection level is required by the consumer for node N

$PPL_{sell}(N)$ – denotes the amount of privacy protection level that can be provided by the provider for node N

$max \sum_i \sum_{N \in Tree_i} v_i(N) x_i(N)$	**(5)**
s.t.	
$lb\ x_i \leq \sum_i \sum_{N' \in child(N)} x_i(N') \leq ub\ x_i(N),\ \ N \in \{Tree_i \backslash leaf(Tree_i)\}$	**(5.1)**
$stime_i\ x_i(N) \leq z_t(N) \leq etime_i\ \ x_i(N),\ \ N \in \{Tree_i \backslash leaf(Tree_i)\}$	**(5.2)**
$PPL_{buy}(N) \geq PPL_{sell}(N),\ N \in \{Tree_i \backslash leaf(Tree_i)\}$	**(5.3)**
$\sum_{N \in leaf(Tree_i)} Q_i(N)\ x_i(N) \leq \lambda_i$	**(5.4)**
$\sum_{N \in leaf(Tree_i)} q_{buy}(N, A_R)\ x_i(N) - \sum_{N \in leaf(Tree_i)} x_i(N)\ \ q_{sell}(N, A_R) \geq 0$	**(5.5)**
$x_i(N) \in \{0,1\}, \lambda_i \in \mathbb{Z}$	**(5.6)**
$z_t(N) \geq 0,\ Q_i(N) \geq 0,\ q_{buy}(N, A_R) \geq 0,\ q_{sell}(N, A_R) \geq 0$	**(5.7)**
$PPL_{sell}(N) \geq 0,\ PPL_{buy}(N) \geq 0$	**(5.8)**

Model 5: Winner Determination Model with the Privacy Concerns.

Constraint (5.1) enforces the interval constraints on the parents' nodes. It ensures that no more and no less than the appropriate number of children is satisfied for any node that is satisfied.

Constraint (5.2) enforces the execution of the nodes is within the required time window described in the parent node.

Constraint (5.3) ensures that privacy protection level concern required by the consumer is met by the provider.

Constraint (5.4) ensures that the quantity of each item across all satisfied leaves is no greater than the total number of units awarded in the trade. This works for providers as well as consumers: for providers a trade is negative, and this requires that the total number of items indicated as sold in the tree be at least the total number of items traded from the bidder in the trade.

Constraint (5.5) ensures the minimum requirements of the consumers are achieved.

6.4 The Winner Determination Algorithm

In this section, we describe the algorithm formed for the winner determination. The main idea of the proposed algorithm is to split the items offered based on the time dimension. For example, a day can be split into four different time windows as follow:

- 12AM to 6AM time window
- 6AM to 12PM time window
- 12PM to 7PM time window
- 7PM to 11:59PM time window

Each time window has a specific bin that stores the bids from consumers and asks from providers. Each bin includes different lists related to the offered items such as, storage, computational resources, and services.

A bin represents set of items to be sold within a specific time window. Bids and asks are introduced into bins based on time requirements. The number of bins in an auction is obtained once the auction is configured.

We classify the bids based on the time bins as:

- Precise bids are the ones that are specific to the time slot.
- Overlapping bids are those that can be introduced or executed on different bins.

6.4.1 Providers' Resource Insertion

Providers' asks can overlap between bins, however, we assume that a provider does not have dependencies between resources in other bins.

The insertion of asks must maintain the social welfare function to be maximized. When an *ask* is inserted in a bin, we check if there are bids within the "losing request list" as shown in Figure 15 that can be matched with the newly inserted *ask*. *Asks* can be matched when inserted.

6.4.2 Consumers' Bids Insertion

When a new bid is received, it is analyzed to be inserted to the proper bin based on its time window classification. The types of bid requests are classified into four types:

1) A request to be executed within a specific bin time window
2) A request to be executed between different bins based on the required time window. For example, a task that is to be executed in bin 1 AND bin 2.
3) A request that can be executed within a specific bin OR a partial schedule that can be executed in bin 1 and the rest of the schedule is executed in bin 2.
4) A request with a specific time requirement that can be executed within a specific bin OR another bin based on specific time requirements and valuation.

Step 1:

When bids are classified to the specific time window, each bid is given a value based on the heuristic score that defines maximum value gain (MVG) by the bid. This is presented by:

$$MVG = \left(\frac{(Bin_{et} - B_{st})}{B_q}\right) * B_v$$

where Bin_{et} presents the bin end time, B_{st} presents the bid start time, B_q is the quantity of the required resources, B_v is the value of the bid. The main purpose of MVG is to find which bin best fits the requested bid, specifically when the requested bid can overlap across multiple bins. If the presented bid has more than one choice, a value is given for each choice. The choice with the greater value is inserted to the proper bin and the other choice is inserted into a Pending Request List as shown in Figure 15. The bids that are in the Pending Request List are retrieved if other choices arrive to the bin that requires an initial resource that was initially matched. The Pending Request List might hold an

alternative choice for the request. If no other choices exist, the optimization algorithm selects the best choice among the selected matches. The pending request list is cleared when the auction clears.

The possible situations that can occur when a bid *B* is inserted are:

- *B* can relocate a winning bid in any of the bins
- *B* can make a current losing ask in any of the bins to be allocated.
- *B* cannot win in any of the bins.

The condition to be maintained is the Social Welfare, so the choice of any of those situations is given by the condition that maximizes the current social welfare. The purpose is to maintain the allocative efficiency when determining the winner.

Step2:

Each bid before it is inserted to the bin is matched with the possible capable resources that can execute the request. If possible resources exist and the bid value is greater than or equal to reserve value, then the request is matched with the possible resources within the bin. If no feasible resources exist or the bid value is less than the providers reserve value, then the request choice is inserted into the losing request list as shown in Figure **15**. Bids are taken out from the losing request list as new resources are inserted to the bin.

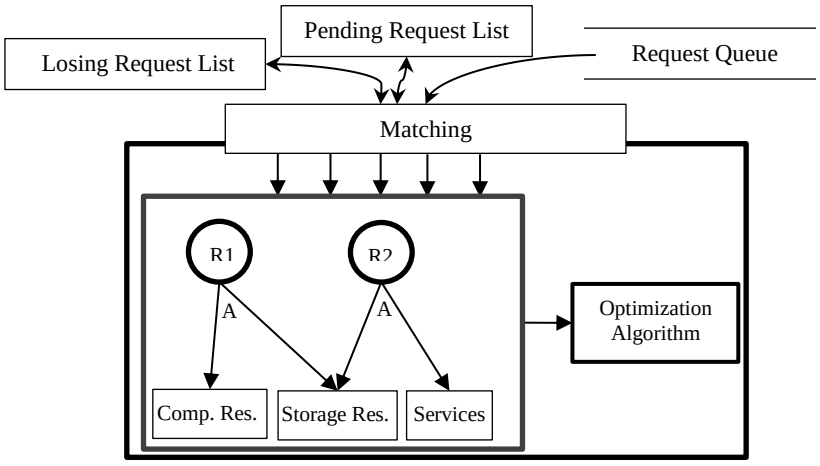

Figure 15: Time-Based Bin Architecture Example.

Step3:

The selection of the resources is based on the MVG by the request. The greatest value is selected from each bin.

The initial allocation from the overlapping bin assigns the specific resources over the bins. After the completion of the overlapping bin allocation, the allocation selection process of the winner of the other bins starts. Similarly, the winner of each bin is selected based on the greatest value of utilizing the resources. The assigned overlapping bids and resources are not included as part of the winner selection process.

Auction mechanism:

The nature of the Grid/Cloud environment eliminates the use of the commonly applied combinatorial auction algorithms (e.g., the Vickrey-Clarke-Groves (VCG) mechanism), as generally such mechanism is not computationally tractable. We utilize a sealed clock auction. All bids are entered within the specific auction time, however, consumers cannot see other bidders valuations, and hence cannot modify their bids based on the actions of others. This is to give the incentive to entities to reveal their truth valuation and not to adjust them based on other entities valuations to the resources. In practice, participants are typically given some window of time in which to enter bids and, possibly, respond to environmental conditions. The auction keeps soliciting bids and asks until the time of the auction ends. The sealed clock auction is computationally tractable. The execution time scales linearly in the number of participants and the number of resources.

Reserve Pricing

The reserve prices form the basis of a decision support framework in the market economy that allows providers to steer the system towards particular, desired outcomes. If one resource pool is particularly crowded, for instance, then the provider can set its reserve price high to ensure that consumers in this pool have the incentive to leave it for another, less crowded one. We use an approach that takes into account the resource loads. For each resource bundle *r*, we assume there is a utilization measure, $\eta(r)$ and that each resource bundle, *r*, has a cost *c(r)*. We then define our reserve price for *r* as:

$$p_{reserve} = w\big(\eta(r)\big) * c(r)$$

where $w(\cdot)$ is the weight function for r. The weight function reflects on the resource availability. The following reflects on the criteria reflects on for constructing these:

- $w(\cdot) > 1$ for resources that are over utilized.
- $w(\cdot) \leq 1$ for resources that are underutilized.
- The relative cost difference of resources in highly congested (e.g. 99% vs 80% utilization) is significantly greater than the cost difference of resources in underutilized (e.g. 40% vs 15% utilization).

The inputs of the weight functions are utilization percentiles for the different resource dimensions (e.g. computation, disk, services).

6.4.3 Auction clear

Auction mechanism for the overlapping bin clears first and the other time-based bins can clear in parallel.

Chapter 7

7 Implementation and Validation

This chapter presents a solution framework for the scheduling problem in the Grid/Cloud computing environment. The framework is based on implementing the proposed bidding language (Chapter 5), and the winner determination approach (proposed in Chapter 6).

In developing practical architectural solutions for complex environments, we propose to model the Grid/Cloud marketplace as software-agent. It is expected that the Grid/Cloud marketplace will include services and participants that involve complex and nondeterministic interactions. These requirements could not be accomplished using traditional ways of manually configuring software. Agent-orientation is a very promising design paradigm for integrating dynamic environment and is essential to model an open environment, such as the Grid/Cloud computing environment.

The main purpose of this chapter is to show the integration of our proposed solution with existing Grid/Cloud technology such as Globus. We also present simulation results related to the quality of the solution and the run time while executing the winner determination with the existence of many bids and asks.

7.1 Proposed Grid/Cloud Scheduling Architecture

The proposed architecture provides a framework for Grid/Cloud entities to integrate with the proposed Grid/Cloud market. There are two main elements to the framework:

- Real-time integration: this component receives information from entities such as the bids, and integrates them with the Grid/Cloud market. It also deals with the entities registration and event handling.
- Grid/Cloud market: this component provides the elements required for the market to enable the auction mechanisms and to manage and configure the lifecycle of the auction. The Grid/Cloud market includes different components that each has a specific role within the framework that corresponds to a market specific functionality.

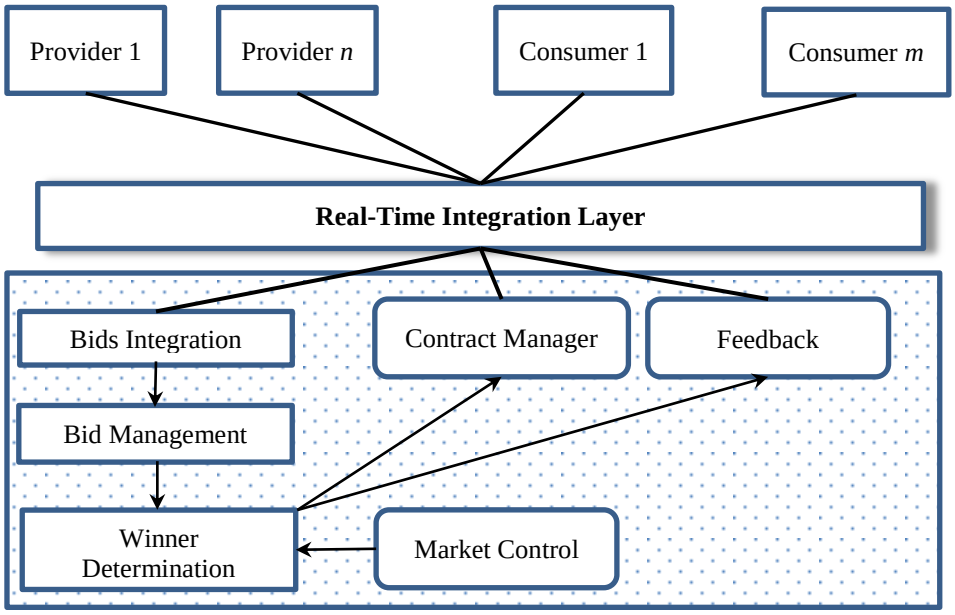

Figure 16: High-level Architecture.

The main components as shown in Figure 16 are:

- Bid Integration: this component integrates the received bids from providers/consumers and integrates it with the winner determination system.
- Bid Management: manages the bids as requests received. As parts of bids are received from consumers, the bid management replaces the bid in the proper place in the bidding language tree. Similarly, removing bids and updating the bidding tree for an entity. It allows pre-processing of incoming bids to match the specific trading conditions of the market.
- Market control: is the main container of the auction and market functionalities. It governs the lifecycle of the market.
- Winner determination: this component implements the auction process and clears the auction. It is triggered by the market control and finds the winner based on the algorithm proposed in Chapter 6.
- Feedback: this component allow participants to listen to market events such as current prices, termination, start of new round and final agreements.
- Contract Manager: handles agreements and facilitates the market clearance.

Components are architected to form the Grid/Cloud market. Specific rules e.g., a new pricing policy can be added to the platform by specializing the relevant component without changing the rest of the architecture.

7.2 Proposed Grid/Cloud Scheduling Architecture with Privacy Concerns

Providers are responsible to supply adequate level of PPL in order to be eligible for executing the consumers' tasks. Similarly, the broker is to provide at least the minimum PPL requirement for consumers and providers to share their requirements and specifications. For instance, when consumers share their task requirements with the broker, they expect to receive enough PPL from the broker in order to disclose the information. Higher number of PPL brings more responsibilities in providers' sides for protecting privacy of consumers. The provided PPL value by an entity indicates the level of privacy protection that the provider is able to provide.

Consumers on the other hand value the minimum required PPL to share its requirements and to execute its tasks. As an expansion for the scheduling solution architecture presented in section 7.1, we added a component that provides privacy matching as shown in Figure **17**. The privacy matching component is to satisfy matching consumers to providers based on the condition $PPL_i x_{ij} \geq PPL_j,\ \forall j \in buy,\ i \in sell$. The results from this component are the possible provider entities that are able to provide the minimum required PPL by the consumer. The result enters the winner determination component in which it finds the allocation of providers to consumers. The winner determination component is the same as proposed in Chapter 6.

With such architecture, the input of the winner determination problem remains unchanged as previously proposed. The addition of the privacy matching component filters the unfeasible space from providers that are unable to achieve the privacy protection level for consumers.

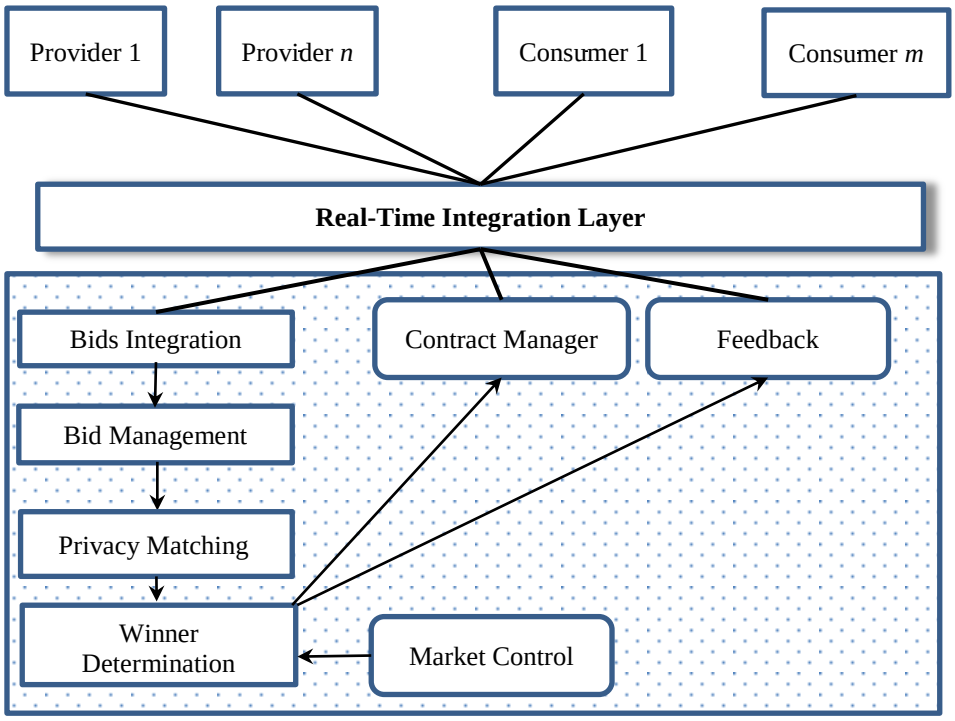

Figure 17: High-level view of the allocation of resource given the privacy concerns.

7.3 Bidding Language Representation

In the implementation of the proposed approach we have selected JSDL for the presenting the proposed bidding language presented in Chapter 5. JSDL (Job Specification and Description Language) is a standard proposed by the Open Grid Forum for describing tasks to be executed on the Grid infrastructure. JSDL is an XML based language. We mapped the proposed bidding language as described in Chapter 5 to the JSDL schema, and used the extensible nature of XML to extend JSDL in order to support the proposed solution.

The scope of the JSDL schema deals with submission requirements of individual tasks only. JSDL specification notes the fact that other documents maybe required to address the entire lifecycle of a task including relationship between other tasks. To support workflow and scheduling requirements, two separate documents are introduced that are JSDL-aware, WSL (Workflow Specification Language) and SDL (Scheduling Description Language).

Workflow support is implemented by introducing the WSL (Workflow Specification Language). WSL has been developed by the C3 project. WSL is JSDL aware. The specification allows for referencing of JSDL elements in order to create dependencies between different tasks.

The time-based requirements in the implementation are described in a separate document as shown in Example 1. This approach is the preferred way to deal with additional parts of the task lifecycle in the Grid environment as described in the JSDL document.

```
<ScheduleDescription>
  <StartTime .../>?
  <EndTime .../>?
</ScheduleDescription>
```

Example 1: Time-based Requirements.

The resource requirements are done through the use of the JSDL core specifications. JSDL has support for both the computational and storage resources as shown in Example 2. The computational requirements of our model map to the elements of the *Resources* tag. Such requirements as CPU speed, number of processors, and memory requirements map to elements within the *Resources* such as *IndividualCPUSpeed*, *IndividualCPUCount*, *IndividualPhysicalMemory*. Data storage requirements also map to similar elements within the *Resources* tag, such as *IndividualDiskSpace*.

```
<Resources>
  <IndividualCPUSpeed>
     <LowerBoundedRange>
       1073741824
     </LowerBoundedRange>
  </IndividualCPUSpeed>
  <IndividualCPUCount>
    <Exact>2</Exact>
  </IndividualCPUCount>
  <IndividualPhysicalMemory>
    <Exact>4G</Exact>
  </IndividualPhysicalMemory>
  <IndividualDiskSpace>
    <Exact>25GB</Exact>
  </IndividualDiskSpace>
  <TotalResourceCount>
    <Exact>2</Exact>
  </TotalResourceCount>
</Resources>
```

Example 2: Resource Requirements.

The *TotalResourceCount* tag gives ability to represent multiple identical units. We further extend the JSDL by adding a multiplicity to the *Resources* tag. The JSDL specifies the multiplicity of the tag to be 1, but in order to satisfy the bidding language we allow for multiple resource requirements to be listed in the task description. This approach allows to further support the interval operator type where the required number of resource to satisfy the task is specified.

```
<JobDefinition>
  <JobDescription>
    <JobIdentification ... />?
    <Application ... />?
    <Resources ... />+
  </JobDescription>
  <ResourceInterval ... />*
</JobDefinition>
```

Example 3: Extended JDSL schema.

We further extend the JSDL to support the interval operator type for the proposed bidding language extension. The interval operator allows a consumer to specify the number of resources which are required to satisfy the task execution. This is implemented using the *ResourceInterval* tag element within the *JobDefinition* element of the JSDL schema. The complete model of our implementation is shown in Example 3. Similarly to the other elements of the *JobDescription* tag, the *ResourceInterval* element has the *jsdl:RangeValue_Type*. The *jsdl:RangeValue_Type* enabled to specification for exact number of resources required to satisfy the task execution, as well, optionally the consumer can specify the upper and/or lower bounds as shown in Example 4. The *jsdl:RangeValue_Type* is defined in the JSDL document. It can contain the following elements *LowerBoundRange*, *UpperBoundRange*, and *Exact*. Where the *LowerBoundRange* denotes the least number of children which are required to satisfy the task execution of the given task, *UpperBoundRange* denotes the most number of children required to satisfy the task execution, and *Exact* denotes the exact number of children required to satisfy the task execution.

```
<ResourceInterval>
  <Exact>2</Exact>
</ResourceInterval>

<ResourceInterval>
  <UpperBoundRange>2</UpperBoundRange>
</ResourceInterval>
```

Example 4: ResourceInterval tag, two examples.

In addition to the implementation of the job definition, we introduce the PPL value. The PPL value is represented as shown in Example 5. The PPL value on the consumer side is submitted with the job definition and represents the required PPL for executing the tasks. The JSDL specification allows for the extension of the attributes for the JobDefinition element. We choose to implement the PPL value as an attribute of the JobDefinition element since the modification of the task definition might not require the change of the PPL attribute.

```
<JobDefinition ... PPL="8">
...
</JobDefinition >
```

Example 5: PPL value implementation.

The provided architecture in section 7.2 allows for the Privacy Matching component within the broker to perform matching of the job definition to the available providers. This approach eliminates any providers which cannot provide enough PPL for consumers.

7.4 Implementation Environment

We developed a prototype of an agent-oriented Grid/Cloud by utilizing Globus toolkit and the Java Agent Development (JADE) platform for the runtime environment as shown in Figure 18. JADE is a software framework which allows us to develop agent applications in compliance with the Foundation of Intelligent Physical Agents (FIPA) specifications for multi-agent systems. JADE deals with all aspects that are external to agents and independent of their applications. These include message transport, encoding,

parsing and agent lifecycles. JADE supports a distributed environment of agent containers, which provide a run-time optimized environment to allow several agents to execute concurrently. This feature has been utilized to create several concurrent auction sessions. A complete agent platform may be composed of several agent containers. Communication in JADE, whether internal to the platform or externally between platforms, is performed transparently to agents. Internal communication is realized using Java Remote Method Invocation to facilitate communication across the Grid/Cloud environment and its market sessions. External non-Java based communication, between the market and its participating organizations, is realized through the Internet InterOrb Interoperability Protocol mechanism or http.

At the resource level, we utilize the functionalities of Globus toolkit version 5.0.2. The use of the Globus technology is limited to task processing and monitoring at each computing node. We use GramJob API within the Java WS Core to provide the necessary methods to submit a task using GRAM and control its lifetime.

In our deployment environment as shown in Figure 18, we have a number of computing nodes connected through the Internet. Each computing node runs Globus Toolkit as the Grid middleware, which provides a uniform access to the computing resource. However from the Globus technology point of view, each Globus computing node is independent of each other and unaware of other existence. On top of the Globus installation we deploy JADE in a distributed configuration.

The Provider agent abstracts each computing node. Each Provider can map a single or multiple computing nodes. Providers are registered in the *Grid* through the Brokering agent. The trading and interaction behavior of the participant agents is governed by the market.

Although our implementation takes advantage of the JADE platform and its supporting agents, such as the directory facilitator (DF), agent management service (AMS), and agent communication center (ACC), the architecture of the application agents is based on the CIR-Agent model [Ghenniwa, 1996]. Java features, such as portability, dynamic loading, multithreading, and synchronization support make it appropriate to implement

the inherent complexity and concurrency for the Grid market. The design of each agent is described in terms of its knowledge and capabilities. The agent's knowledge includes the agent's self-model, goals, and local history of the world, as well as a model of its acquaintances. The agent's knowledge also includes its desires, commitments, and intentions as related to its goals.

The main capabilities of the CIR-Agent include communication, reasoning, and domain actions. Implementation of the communication component takes advantage of JADE messaging capabilities. It is equipped with an incoming message inbox, whereby message polling can be both blocking and non-blocking, and with an optional timeout mechanism. Messages between agents are based on the FIPA ACL. The agent's reasoning capabilities include problem solving and interaction devices.

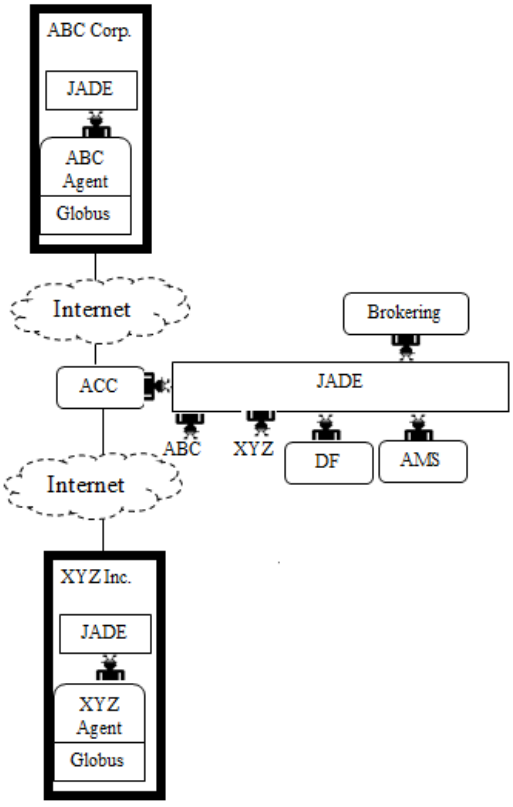

Figure 18: Implementation Logical Architecture.

The brokering agent creates a scheduling decision based on the interaction between consumer and provider agents. As proposed in Chapter 6, the broker-agent interaction and problem solver components are implemented as follows:

- Interaction – it describes the interaction protocol used by the broker-agent to coordinate with the consumer and provider-agents in the environment. The interaction component of the broker-agent is implemented by making use of the existing JADE behavior classes: *FipaContractNetIntitiatorBehavior*. In that protocol, the broker-agent can solicit proposals from consumers and providers by sending a *CFP (call for proposal)* message. Consumers send the bidding to the required items and providers submit the reserve value for their resources. The *PROPOSE* messages, sent by the providers and consumers are taken into the broker-agent problem solver and an *OFFER* message is created to the winner provider to schedule the request.
- Problem Solver – it is the decision making of the broker agent to schedule requests into the resources, based on its self-knowledge, and the knowledge of the provider and consumer agents. The architecture within the broker agent is the proposed architecture in Figure 16. The broker agent announces the winner based on the mechanism proposed in Chapter 6.

The role of the consumer-agent is to express its preferences through the proposed biding language in Chapter 5 format and sends "bid" messages out to the broker-agent. The interaction of the consumer-agent interaction and problem solver is as follows:

- Interaction – describes the interaction protocol used by consumer-agents to interact with the broker-agent in the environment. It contains a class that extends the JADE behavior class *ContractNetReponder* Behavior through which the consumer-agent prepares the PROPOSE message that is later followed by the formulated biding language proposed in Chapter 5 using the JSDL standard.
- The problem-solver – contains a Bid class that implements a cyclic behavior in order to respond to incoming messages from the broker-agent that requests bids. This class implements all the consumer-agent's tasks such as registration with the broker agent as well as a method that formulates preferences and bid valuations using the proposed

bidding language in Chapter 5. The bidding language is created based on the consumer agent objective.

The provider-agent's role is to express the resource specifications and reserve value through the bidding language proposed in Chapter 5. The provider-agent's reasoning component consists of the following:

- Interaction – describes the interaction protocol used by provider-agents to interact with the broker-agent in the environment. The resource's interaction makes use of the existing JADE class *FipaContractNetResponderBehavior* when interacting with the broker-agent.
- Problem Solver – contains *Ask* class that implements a cyclic behavior in order to respond to incoming messages from the broker-agent. This class implements all the provider-agent's requirements such as registration with the broker agent as well as a method that formulates the *Asks* and reserve values using the proposed bidding language in Chapter 5 using the JSDL standard. The reserve values are created based on the provider agent objective.

7.5 Experimentation Environment and Results

The aim of the experiment is twofold: to validate that the winner determination provides quality of the solution and to show the runtime of the proposed work. To carry out the experiments a set of random data sets have been generated. We generated the set size to be as realistic as the size of a real environment. The problem set consists of the following:

Set 1 – consists of 300 bids and 300 asks

Set 2 – consists of 400 bids and 300 asks

Set 3 – consists of 500 bids and 300 asks

Set 4 – consists of 600 bids and 300 asks

Set 5 – consists of 700 bids and 300 asks

Set 6 – consists of 800 bids and 300 asks

Set 7 – consists of 900 bids and 300 asks

Set 8 – consists of 1000 bids and 300 asks

Set 9 – consists of 1100 bids and 300 asks

The distribution functions used were derived from several experiments found in the literature [Mills and Dabrowski, 2008][Phelps, 2007] to generate random data. Uniform distribution of ask prices are motivated by the assumption that costs of resources are also uniformly distributed. Bid prices have been generated using Uniform distribution. The bids and asks are distributed across time slots.

7.5.1 Economic Efficiency

The experiment aimed to evaluate the economic efficiency obtained by the winner determination algorithm. Economic efficiency is defined as the social welfare that the mechanism provides given a certain input. In order to evaluate the economic efficiency of the proposed winner determination, we compared the outcome of the proposed algorithm with CPLEX 10 by implementing the winner determination model created in Model 4 in Chapter 6. We generated 9 random runs for the experiment. For each run the random input is stored and transformed to be used as the input for the CPLEX tool to avoid divergences due to randomization.

We measured the efficiency of Scheduling, $efficiency(S)$, as the ratio of the value of the final schedule S to the value of the optimal schedule provided by CPLEX that maximizes total value across the agents as defined in the model in section 6.2:

$$efficiency(S) = \frac{\sum_{g \in N, S^g \in S} v^g(S^g)}{\sum_{g \in N} v^g(S^*)} * 100$$

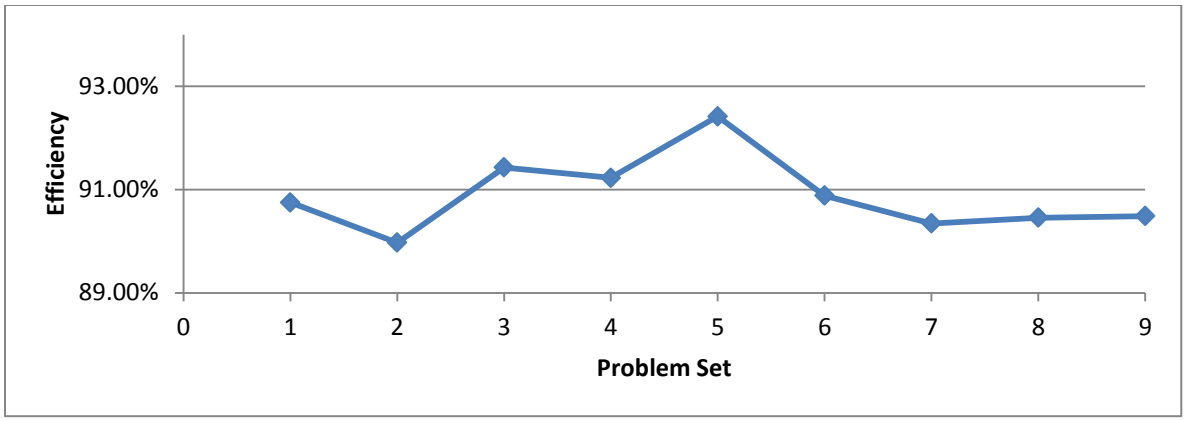

Figure 19: Economic Efficiency for 9 random runs.

The results reflects the difference between the proposed winner determination algorithm that solicits bids based on the time auction mechanism, and the one-shot using CPLEX in the context of auction-based decentralized scheduling.

Figure 19 plots the economic efficiency of the proposed winner determination over the 9 random problem sets. Compared to CPLEX tool which provides (100% efficiency), on average, the proposed winner determination can on average achieve more than 90% efficiency.

We reflect on the mechanism result behavior on the economic efficiency based on two elements: the deficiency by average 10% of the solution quality, and on the average 90% efficiency. It was noticeable that giving priority to the allocation of the overlapping bin created some deficiency to the overall solution. As it created specific solutions that overlap across bins, some other solution were not accommodated within the bins because of the priority given to the overlapping bin. However, because of the MVG heuristic function, the allocation was still controlled not to give full priority to the overlapping bin without having sufficient valuation to the bid. The MVG heuristic, managed to maintain the economic efficiency of the solution. It was also noticeable that the losing request list in the architecture helped in the economic efficiency by not completely ignoring the bids that were not initially matched to asks because of insufficient valuation or capability existence. As new resources (asks) arrive to the environment, bids were taken out from the list. Moreover, the reserve value made the system eliminate the solution space that did

not meet the minimum valuation of asks. This is also a factor that contributed to the economic efficiency within the solution.

7.5.2 Run Times Results

We did a comparison with the run time between the proposed algorithm and CPLEX as the problem size grows through 9 random generated problem sets with 4 time-based bins. The experiment were conducted on an i7-2600, 3.40GHz with 8GB memory. We can see through Figure 20 that the proposed winner determination for the Grid/Cloud environment requires less time to solve a set of problem than CPLEX.

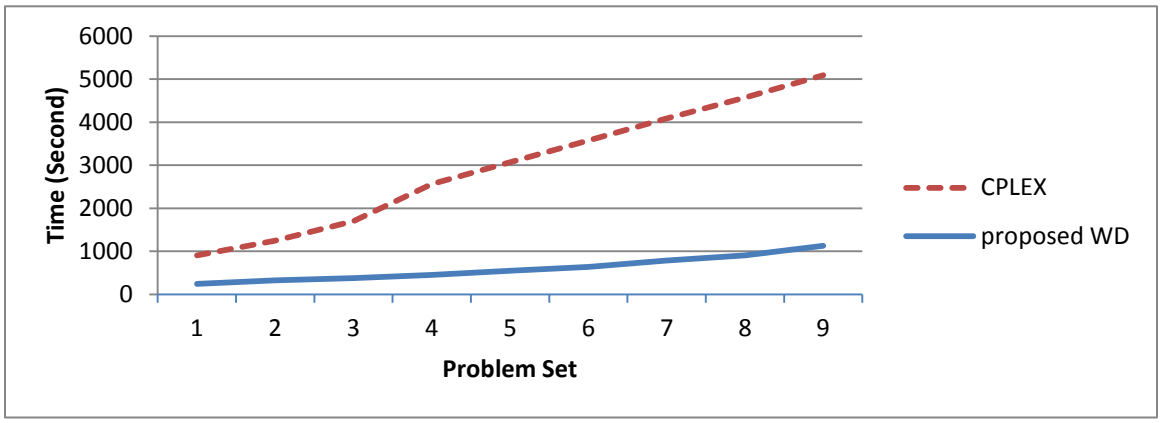

Figure 20: Run times of the propose WD and CPLEX for 9 problem sets.

Couple of factors of the proposed mechanism enabled the behavior of time result. The choice of the clock auction is computationally tractable and limited participants in the auction to provide their preferences within specific time frame. Moreover, splitting the bids and asks into bins split the large problem into small sub-problems. In this experiment case, the problem was split into four sub-problems where the sub problems are executed in parallel. In the worst case scenario, this creates four times more efficient solution than CPLEX. Moreover, the matching process happen as the auction is soliciting bids and asks during the specific auction time. The matching component cuts from the infeasible space of the solution and hence, the solution that are computed by the close of the auction are the once within the feasibility space for each bin.

Chapter 8

8 Summary and Conclusion

This thesis investigates modeling and computational issues in developing solution approaches to the Grid/Cloud scheduling problem. Our objective is to design economic-based models capable of coordinating the scheduling behaviors of independent entities in the Grid/Cloud. The developed solution mainly targets to the valuation, communication, and winner determination complexities in auction-based decentralized scheduling that is adequate for the Grid/Cloud. This chapter summarizes the main contributions of this work; highlights our conclusions; and presents some future research directions.

8.1 Summary of Contributions

In the design of economic-based models for the Grid/Cloud scheduling problem, we focused on auction-based approaches. We addressed complexity issues in applying combinatorial auctions to the Grid/Cloud scheduling problem. Our main contributions include the Grid/Cloud scheduling problem model and analysis, bidding language, winner determination model and algorithm design, and auction structure design in which they are adequate for the Grid/Cloud.

Grid/Cloud scheduling problem modeling: A formal Grid/Cloud scheduling problem model is presented. This model extends the classical centralized scheduling problem for the makespan and resource utilization models to decentralized Grid/Cloud environment. The model was analyzed and derived based on the structure and the characteristics of the Grid/Cloud. A formal mapping to combinatorial auction problems is provided. Comparing with other research work on the Grid/Cloud scheduling problem modeling, our model is more formal and more comprehensive.

Tree-based Requirement Specification Bidding Languages: The proposed language use requirements for processing a set of tasks as atomic propositions and prices are attached to the completion times of the processing and on the specification of the required resources. The requirement specification extension is based on the TBBL language

proposed by [Cavallo et. al. 2005] and designed specifically to suite the Grid/Cloud scheduling problem that applies the auction-based mechanism. This has advantages over other general logical languages in terms expressiveness for entities in the Grid/Cloud, reduces valuation and communication complexities.

Winner Determination model and algorithm: The winner determination problem formulated using the tree-based requirement specification bidding language. We utilized a sealed bid clock auction mechanism to bound the solicitation of bids based on specific time window that is known to all users. Our mechanism splits the problem into sub problems to reduce the complexity of the overall problem. We built an architecture that manages the bids as they arrive and eliminates infeasible space from the scheduling problem. The uniqueness of this framework is it targets winner determination problems formulated by tree-based requirement specification language and targets properties related to strategy-proofness, individual rationality, time efficiency, and economic efficiency. Compared with general winner determination algorithm used in optimization, our approach demonstrates improved performance.

By embedding the tree-based requirement specification languages and winner determination algorithm that solicit bids within specific time frame enables entities to provide their bidding iteratively. This is more natural in terms of the implementation in real world Grid/Cloud.

Scheduling with privacy concerns model and architecture for the Grid/Cloud: We argue in this work that privacy is a requirement component to consider in the decision of the scheduling problem in the Grid/Cloud given the nature of the environment. There is very little work done in this domain. In this thesis, we analyzed and developed a scheduling model that takes privacy concerns of entities within the scheduling problem. We have expanded our Grid/Cloud computing scheduling model, mechanism, and architecture to enable the privacy concern in the scheduling decision for the Grid/Cloud.

8.2 Conclusions

Auctions offer great promise as mechanisms for optimal resource allocation in the Grid/Cloud environment. However, the applicability of auctions to the Grid/Cloud scheduling depends on the ability to manage the valuation, communication, winner determination, and strategic complexities in the context of scheduling problem. We argue that it is necessary to take an explicit computational approach, which integrates scheduling specific methods, to auction-based Grid/Cloud scheduling system design. The proposed bidding language presented in this thesis is designed for the Grid/cloud scheduling problem. We have shown that the proposed bidding language provides concise, natural representations of entities' valuations for the Grid/Cloud scheduling. In addition, the winner determination problem resulted from the languages preserve natural scheduling constraints which enables effective algorithm design. We developed the winner determination algorithm which embeds constraint-directed search scheduling. The experimental results have exhibited significant improvement in terms of problem solving speed and maintain the economic efficiency to at least 90%.

8.3 Directions for Future Research

This thesis improves on the understanding on the modeling of the scheduling problem in the Grid/Cloud environment and advances the state-of-the-art through its contributions. The investigations conducted in this thesis reveal several areas in Grid/Cloud scheduling, where much work remains to be done. Moreover, the contributions of this thesis have led to new challenges that are to be addressed through further research. This section briefly describes some of these challenges within the scope of the thesis.

First, we will continue to improve on the efficiency of the winner determination. The proposed algorithm developed in the thesis has demonstrated good performance in auction-based Grid/Cloud scheduling. Heuristics from classical scheduling theory can be embedded to boost the approach's performance on well-studied scheduling problem models for each bin. We will explore the possibility of introducing approximate and heuristic algorithms for the winner determination problem. While these algorithms can

come with different flavors, those preserve incentive compatibility are worth of investigation.

Second, we will continue the investigation on the privacy concerns in the Grid/Cloud specifically into the quality of the solution aspect. Our proposed economic-based model deals with privacy as a quality measure, however, in our proposed architecture, we created a component that deals with matching requests to resources that are able to provide at least the minimum privacy protection level. The proposed heuristic does not guarantee that entities are receiving the maximum privacy protection level from the Grid/Cloud resources. Future expansion is to measure the quality of the proposed privacy heuristic solution quality and investigate into possible improvement to the scheduling solution given privacy concerns in the Grid/Cloud.

A third direction is to investigate models to include energy-aware resource allocation qualities. There is a growing demand for computational power from industry and academia that has led to extreme power consumption. Numbers of initiatives were taken in the development of energy-efficient hardware. The overall energy consumption however, continues to grow due to the overwhelming requirements for computing resources and data centers. Utilizing the consumption of the power in an inefficient way will eventually lead to critical problems such as, insufficient or malfunctioning to the cooling system. This result to overheating of resources and reduces the system reliability and lifetime. Moreover, high power consumption leads to generating substantial amount of carbon dioxide. The proposed architecture in this thesis considers the scheduling decision based on time, resource utilization, and privacy concerns. Further direction is to extend the scheduling model with energy-aware resource allocation that takes into account both consolidation (to switch off nodes) and smart task mapping techniques with a view to lower the total energy consumed to run a service.

A fourth direction is to enhance the reliability of critical tasks execution. A task in the Grid/Cloud is called critical task if the execution of other tasks depend on the output or the execution completion of the “critical” task. The failure of executing a task in the Grid/Cloud can be caused by changes in the resources environment configuration, non-

availability of required services or software components, overloaded resource conditions, and faults in computational and network fabric components. Proposed techniques to achieve fault-tolerance in [Abawajy, 2004] such as retry, check-pointing, and redundant task-allocation. The redundant task-allocation technique [Abawajy, 2004] executes the same task simultaneously on different resources to guarantee fault-tolerant execution of that task in the event of task failure, provided that one of the resources does not fail. It is not efficient to apply redundant task-allocation for each task in a workflow, rather it can be applied for critical tasks. The challenges along the future research direction is to look into improving reliability of workflow execution in case of unexpected resource behavior in the Grid/Cloud and to develop algorithms for identifying the critical tasks and determining the level of redundancy based on the reliability requirement.

Bibliography

[Abawajy, 2004] Abawajy J. H. Fault-tolerant scheduling policy for grid computing systems. In Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04), USA, April, 2004.

[Aderholz et al.,2001] Aderholz, M. et al. (2001). MONARC Project Final Report. Technical report, CERN. URL: http://cern.ch/lhc-computing-review-public/Public/Report_final.PDF

[Andersson et al., 2000] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for auctions with bids for combinations. In Proc. 4th International Conference on Multi-Agent Systems (ICMAS-00), 2000.

[Bagchi and Uckum, 1991]Bagchi S., Uckum S. (1991). Exploring Problem-Specific Recombination Operators for Job shop Scheduling. Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, San Diego.

[Bartal et. al., 2003] Y Bartal, R Gonen, and N Nisan. Incentive compatible multi unit combinatorial auctions. Technical report, The Hebrew University of Jerusalem, 2003.

[Benisch et. al., 2008] M. Benisch, N. Sadeh, and T. Sandholm, "Theory of expressiveness in mechanisms", in AAAI-08, 17-23, 2008

[Berman et al. 1997] Berman F. and Wolski R., (May 1997). "The AppLeS Project: A Status Report", Proceedings of the 8th NEC Research Symposium, Berlin, Germany.

[Berman et al., 2003] Berman F., Wolski R., Casanova H., Cirne W., Dail H., Faerman M., Figueira S., Hayes J., Obertelli G.,. Schopf J, Shao G., Smallen S., Spring N., Su A. and Zagorodnov D., Adaptive Computing on the Grid Using AppLeS, in IEEE Trans. On Parallel and Distributed Systems (TPDS), Vol.14, No.4, pp.369--382, 2003.

[Bezzi, 2010] Bezzi M. (2010). "An Information Theoretic approach for privacy metrics", Transactions on Data Privacy 3, p.199-215.

[Bikhchandani and Ostroy, 2006] Bikhchandani S. and Ostroy J. M. “Ascending price Vickrey auctions,” Games and Economic Behavior, Volume: 55, Issue: 2, May, 2006, pp. 215-241.

[BIRN, 2005] Biomedical Informatics Research Network (BIRN) (2005). http://www.nbirn.net.

[Blumrosen and Nisan, 2002] Liad Blumrosen and Noam Nisan. Auctions with severely bounded communication. In Proc. 43rd Annual Symposium on Foundations of Computer Science, 2002.

[Blythe et. al., 2005] Blythe J., Jain S., Deelman E., Gil A., and Vahi K.. Task scheduling strategies for workflow-based applications in grids. In Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid’05), UK, May, 2005.

[Boutilier, 2002] Craig Boutilier. Solving concisely expressed combinatorial auction problems. In Proc. 18th National Conference on Artificial Intelligence (AAAI-02), July 2002.

[Briscoe and Marinos, 2009] Briscoe, G., and A. Marinos. "Digital Ecosystems in the Clouds: Towards Community Cloud Computing." Digital Ecosystems and Technologies Conference. IEEE Press, 2009.

[Bulhoes et al., 2004] Bulhoes, P. T., Byun, C., Castrapel, R., and Hassaine, O. (2004). N1 Grid engine 6 features and capabilities. White paper, Sun Microsystems, Phoenix, USA.

[Buskens, et al, 2000] Buskens, Vincent and Jeroen Weesie (2000), “Cooperation via Social Networks,” Analyse and Kritik, this issue.

[Buyya, 2002] Buyya, R. “Economic Paradigm for Distributed Resource Management and Scheduling for Service Oriented Grid Computing,” Ph.D. thesis, Monash University, April 12, 2002.

[Buyya et al.,2000a] Buyya R., Abramson D., and Giddy J., (May 2000). “Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid”, The 4th International Conference on High Performance

Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China, IEEE Computer Society Press, USA.

[Buyya et al.,2000b] Buyya R., Giddy J., Abramson D., (August 2000). "An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications", Proceedings of the 2nd International Workshop on Active Middleware Services (AMS 2000), Kluwer Academic Press, Pittsburgh, USA.

[Buyya et al., 2002] Buyya R., Abramson D., Giddy J., and Stockinger H., (2002). "Economic Models for Resource Management and Scheduling in Grid Computing", The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press.

[Buyya et al., 2001] Buyya R. and Vazhkudai S., (2001). "Compute power market: Towards a market-oriented grid", The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001).

[Buyya et. al., 2008] Buyya R., Yeo C. S., and Venugopal S., "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", Keynote Paper, Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008, IEEE CS Press, Los Alamitos, CA, USA), Sept. 25-27, 2008, Dalian, China.

[Casanova et al., 2000] Casanova H., Legrand A., Zagorodnov D. and Berman F., (May 2000). Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, in Proc. of the 9th hetero-geneous Computing Workshop (HCW'00), pp. 349-363, Cancun, Mexico.

[Cavallo et. al. 2005] Cavallo, R., Parkes, D. C., Juda, A. I., Kirsch, A., Kulesza, A., Lahaie, S., Lubin, B., Michael, L., & Shneidman, J. (2005). TBBL: A Tree-Based Bidding Language for Iterative Combinatorial Exchanges. In Multidisciplinary Workshop on Advances in Preference Handling (IJCAI).

[Compte and Jehiel, 2000] Olivier Compte and Philippe Jehiel. On the virtues of the ascending price auction: New insights in the private value setting. Technical report,

CERAS and UCL, 2000.

[Condon et al., 1967] Condon WS, Ogston WD (1967) A segmentation of behavior. J Psychiat Res 5:221–235

[Condor, 2012] Condor High Throughput Computing - http://www.cs.wisc.edu/condor (visited December 2012)

[Conway et. al., 1967] Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967) Theory of Scheduling. Addison-Wesley, Reading, Mass.

[Cooper et al., 2004] Cooper K., Dasgupta A., Kennedy K., Koelbel C., Mandal A., Marin G., Mazina M., Mellor-Crummey J., Berman F., Casanova H., Chien A., Dail H., Liu X., Olugbile A., Sievert O., Xia H., Johnsson L., Liu B., Patel M., Reed D., Deng W., Mendes C., Shi Z., YarKhan A. and Dongarra J., (April 2004). "New Grid Scheduling and Rescheduling Methods in the GrADS Project". In Proceeding of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), pp.199--206, Santa Fe, New Mexico USA.

[Czajkowski, 1998] Czajkowski K., Foster I., Karonis N., Kesselman C., Martin S., Smith W., and Tuecke S., (March 1998). "A Resource Management Architecture for Metacomputing Systems", In D.G.Feitelson and L. Rudolph, editors, in Proc of the 4th Workshop on Job Scheduling Strategies for Parallel Processing, LNCS Vol. 1459 pp. 62–82, Orlando, Florida USA.

[Czajkowski,2001] Czajkowski K., Fitzgerald S., Foster I., and Kesselman C., (August 2001). "Grid Information Services for Distributed Resource Sharing", in Proceeding the 10th IEEE International Symposium on High- Performance Distributed Computing (HPDC-10), pp. 181-194, San Francisco, California, USA.

[Davis, 1985] Davis L., (1985). Job Shop Scheduling with Genetic Algorithms. Proceedings of an International Conference on Genetic Algorithms and their Applications, Pittsburgh, Lawrence Erlbaum Associates.

[Dey et al., 2002] A. Dey , S. Lederer , J. Mankoff , "A Conceptual Model and Metaphor of Everyday Privacy in Ubiquitous Computing", Technical report, University of California at Berkley, USA, 2002.

[De Vries and Vohra, 2002] Sven de Vries and Rakesh V Vohra. Combinatorial auctions: A survey. Informs Journal on Computing, 2002.

[Doebeli, et al, 2005] Doebeli, M. and Hauert, C. “Models of cooperation based on the Prisoner’s Dilemma and the Snowdrift game.” Ecology Letters. 8 (2005) 748-766.

[Durfee et al., 1989] Durfee, E., Lesser, V. and Corkill, D., (1989). “Cooperative distributed problem solving,” In A. Barr, P. Cohen, and E. Feigenbaum, editors, The Handbook of Artificial Intelligence, volume IV, pages 83-147, Addison Wesley.

[Ernemann et al., 2005] Ernemann C. and Yahyapour R., (2005). “Grid Resource Management: State of the Art and Future Trends”, chapter 30 Applying Economic Scheduling Methods to Grid Environments, pages 491–506.

[Fahringer et. al., 2005] Fahringer T., Jugravu A., Pllana S., Prodan R., Seragiotto C., and Truong H. L. Askalon: A tool set for cluster and grid computing. Concurrency and Computation: Practice and Experience, vol. 17, no. 2-4, pp. 143-169, 2005.

[Fisher, 1973]Fisher, M. L., (1973) Optimal solution of scheduling problems using Lagrange multipliers: Part I. Operations Research, 21:1114-1127.

[Foster, 1998] Foster, I. Computational Grids, pp. 15–52. In [10], 1998.

[Foster and Kesselman, 1998] Foster, I. and Kesselman, C. The Globus Project: a Status Report. In Proc. IPPS/SPDP’98 Workshop on Heterogeneous Computing, pp. 4–18, 1998.

[Foster et al. 2002] Foster, I., Kesselman, C., Nick, J., and Tuecke, S. Grid Services for Distributed System Integration. Computer, 35(6):37–46, 2002.

[Foster et al., 2001] Foster, I., Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. Supercomp. App., 15(3):200–222, 2001.

[Foster et al., 2008] Foster I., Zhao Y., Raicu I., and Lu S., Cloud Computing and Grid Computing 360-Degree Compared, Grid Computing Environments Workshop, 2008. (GCE '08), 1-10, Austin, Texas, USA, November 2008.

[Franks, 1989] Franks, N., (1989). "Army Ants: A Collective Intelligence", American

Scientist, vol. 77, pp.139-145.

[French, 1982] French, S., (1982) Sequencing and Scheduling' An Introduction to the Mathematics of the Job Shop, New York: John Wiley and Sons, inc.

[Frey et al., 2001] Frey, J., Tannenbaum, T., Livny, M., Foster, I. T., and Tuecke, S. (2001). Condor-G: A computation management agent for multi-institutional Grids. In 10th IEEE International Symposium on High Performance Distributed Computing (HPDC 2001), pages 55–63, San Francisco, USA. IEEE Computer Society.

[Garey and Johnson, 1979] Garey, M.R., and Johnson, D. S. 1979. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman Company.

[Geelan, 2009] Geelan Jeremy. Twenty one experts dene cloud computing. Virtualization, January 2009. Electronic Magazine, article available at http://virtualization.sys-con.com/node/612375.

[Ghenniwa, 1996] Ghenniwa H., Coordination in cooperative distributed systems, Ph.D. Thesis, University of Waterloo, 1996

[Giffler, 1960]Giffler B. and Thompson G. L. (1960). Algorithms for solving production scheduling problems, Operations Research, 8, 487-503.

[Glover, 1986]Glover F., (1986). Future paths for integer programming and links to artificial intelligence, Computers & Operations Research, 5: 533-549.

[Greenstadt, 2008] R. Greenstadt. An analysis of privacy loss in k-optimal algorithms. In Workshop on Distributed Constraints Reasoning (DCR08), at AAMAS 2008, Estoril, Portugal, May 2008.

[Greenstadt et. al., 2006] R. Greenstadt, J. Pearce, M. Tambe, "Analysis of Privacy Loss in Distributed Constraint Optimization", AAAI'06 Proceedings of the 21st national conference on Artificial intelligence - V 1, pp 647-653 , 2006.

[Grimshaw, 2002] Grimshaw, A. What is a Grid? Grid Today, 1(26), 2002.

[Grimshaw and Wulf, 1997] Grimshaw, A. and Wulf, W. The Legion Vision of a Worldwide Virtual Computer. Comm. of the ACM, 40(1):39–47, 1997.

[Groves, 1973] Groves Theodore. Incentives in teams. Econometrica, 41:617-631, 1973.

[Hamscher,2000] Hamscher V., Schwiegelshohn U., Streit A., Yahyapour R., (December 2000). “Evaluation of Job-Scheduling Strategies for Grid Computing”. In Proceeding of GRID 2000 GRID 2000, First IEEE/ACM International Workshop, pp. 191-202, Bangalore, India.

[He et al. 2003] He X., Sun X. and Laszewski G., (July 2003)A QoS Guided Min-Min Heuristic for Grid Task Scheduling, in J. of Computer Science and Technology, Special Issue on Grid Computing, Vol.18, No.4,pp.442—451

[Holzman et. al., 2001] R Holzman, N Kfir-Dahav, D Monderer, and M Tennenholtz.Bundling equilibrium in combinatorial auctions. Games and Economic Behavior, 2001.

[Kalagnanam and Parkes, 2004] Kalagnanam, J, Parkes, D. (2004). “Auctions, Bidding and Exchange Design,” in Simchi-Levi, Wu, Shen, Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era, Kluwer Academic Publishers.

[Kamel and Ghenniwa, 1995] M. Kamel and H. Ghenniwa, 1995, “Partially-Overlapped Systems: The Scheduling Problem,” in *Design and Implementation of Intelligent Manufacturing Systems*, Parsaei, H. and Jamshidi, M. (Eds.), Prentice-Hall, pp. 241-274.

[Kirkpatrick et. al, 1983]Kirkpatrick, S., Gelatt, C. D., (1983). Optimization by Simulated Annealing, Science 220:671-680.

[Krauter et. al., 2002] Krauter, K., Buyya, R., and Maheswaran, M. A taxonomy and survey of grid resource management systems for distributed computing. Int. J. of Software Practice and Experience, 32(2):135–164, 2002.

[Kurzban, 2001] Kurzban, R., The Social Psychophysics of Cooperation: Nonverbal Communication in a Public Goods Game, J. Nonverbal Behav. 25 (2001), pp. 241–259.

[Lehmann, et. al., 2002] Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. Journal of the ACM, 49(5):577-602, September 2002.

[Lehmann et. al., 2006] D. Lehmann, R. Muller, and T. Sandholm, "The Winner

Determination Problem," in Combinatorial Auctions, P. Cramton, Y. Shoham, and R. Steinberg, Eds., MIT Press, 2006.

[Li and Baker 2005] Li M. and Baker M., The Grid: Core Technologies, Wiley, 2005, pp. 274-278

[Litzkow et al. 1988] Litzkow M., Livny M., and Mutka M., (1988). “Condor - A Hunter of Idle Workstations”, Proceedings of the 8th International Conference of Distributed Computing Systems (ICDCS 1988), January 1988, San Jose, CA, IEEE CS Press, USA.

[Lubin et. al., 2008] B. Lubin, A. Juda, R. Cavallo, S. Lahaie, J. Shneidman, and D. Parkes, “ICE: An Expressive Iterative Combinatorial Exchange”, J. of Artificial Intelligence Research, 33, pages 33-77, 2008.

[Mandal et. al., 2005] Mandal Anirban, Kennedy Ken, Koelbel Charles, Marin Gabriel, Mellor-Crummey John, Liu Bo, Johnsson Lennart. Scheduling strategies for mapping application workflows onto the grid. In Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC’05), USA, July, 2005.

[Mas-Colell et al., 1995] Mas-Colell Andreu, Whinston Michael, and Green Jerry R. Microeconomic Theory. Oxford University Press, New York, 1995.

[Maheswaran et. al.,1999] Maheswaran M., Ali S., Siegel H.J., Hensgen D., and Freund R.. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In Proceedings of the 8th Heterogeneous Computing Workshop (HCW’99), Puerto Rico, April, 1999.

[McFedries, 2008] McFedries Paul. The cloud is the computer. IEEE Spectrum Online, August 2008. Electronic Magazine, available at http://www.spectrum.ieee.org/aug08/6490.

[Mills and Dabrowski, 2008] Kevin L. Mills and Christopher Dabrowski. Can economics-based resource allocation prove effective in a computation marketplace? Journal of Grid Computing, 6:291-311, Sept 2008.

[Nagaratnam et al., 2002] Nagaratnam N., Janson P., Dayka J., Nadalin A., Siebenlist

F., Welch V., Foster I., and Tuecke S., "The Security Architecture for Open Grid Services," Open Grid Service Architecture Security Working Group, Global Grid Forum, 2002.

[Nash, 1950] Nash John. Equilibrium points in n-person games. In Proceedings of the National Academy of Sciences, volume 36, pages 48-49, 1950.

[Nisan, 2000] Noam Nisan. Bidding and allocation in combinatorial auctions. In Proc. 2nd ACM Conf. on Electronic Commerce (EC-00), pages 1-12, 2000.

[Nisan and Ronen, 2000] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In Proc. 2nd ACM Conf. on Electronic Commerce (EC-00), pages 242-252, 2000.

[Nisan and Ronen, 2001] Noam Nisan and Amir Ronen. "Algorithmic mechanism design". Games and Economic Behavior, 35:166-196, 2001.

[Nisan and Segal, 2002] Noam Nisan and I Segal. The communication complexity of efficient allocation problems. Technical report, Hebrew University and Stanford University, 2002.

[Nishi, et. al. 2004] Nishi, Tatsushi; Konishi, Masami; Hasebe, Shinji (2004). A decentralized scheduling method for flowshop problems with resource constraints Electrical Engineering in Japan Volume: 149, Issue: 1, October 2004, pp. 44 – 51

[Openpbs, 2012] PBS Works -- http://www.pbsworks.com/ (visited December 2012)

[Parkes, 1999a] David C Parkes. "Optimal auction design for agents with hard valuation problems". In Proc. IJCAI-99 Workshop on Agent Mediated Electronic Commerce, pages 206-219, July 1999. Stockholm

[Parkes, 1999b] David C Parkes. iBundle: An efficient ascending price bundle auction. In Proc. 1st ACM Conf. on Electronic Commerce (EC-99), pages 148-157,1999.

[Parkes, 2001] David C Parkes. "Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency". PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001.

[Parkes and Kalagnanam, 2005] Parkes D. C. and Kalagnanam J., "Models for Iterative

Multiattribute Procurement Auctions," Management Science, 51-3, pp. 435-451, Mar, 2005.

[Parkes and Ungar, 2001] Parkes D. C. and Ungar L., "An Auction-Based Method for Decentralized Train Scheduling. In the Proceedings of 5th International Conference on Autonomous Agents (AGENTS-01), Montreal, Canada, pp. 43-50, 2001.

[Pearce, 1986] David W. Pearce. The MIT dictionary of modern economics. MIT Press, 1986.

[Phelps, 2007] Steve Phelps. Evolutionary Mechanism Design. Ph.D thesis, University of Liverpool (U.K.), 2007.

[Rahman et. al., 2007] Rahman M., Venugopal S., and Buyya R.. A dynamic critical path algorithm for scheduling scientific workflow applications on global grids. In Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (eScience'07), India, December, 2007.

[Ranjan et. al., 2008] Ranjan R., Rahman M., and Buyya R. A decentralized and cooperative workflow scheduling algorithm. In Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'08), France, May, 2008.

[Reeves et. al., 2005] Reeves, D. M., M. P.Wellman, J.K.Mackie-Mason, and A.Osepayshvili. "Exploring bidding strategies for market-based scheduling". Decision Support Systems, 39: 67–85, 2005.

[Ronen, 2001] Amir Ronen. Mechanism design with incomplete languages. In Proc. 3rd ACM Conf. on Electronic Commerce (EC'01), 2001.

[Rothkopf et. al., 1998] Michael H Rothkopf, Aleksandar Pekec, and RonaldMHarstad. Computationally manageable combinatorial auctions. Management Science, 44(8):1131-1147, 1998.

[Rutherford, 1992] Donald Rutherford. Dictionary of economics. Routledge, 1992.

[Sacerdoti et al., 2003] Sacerdoti F.D.,. Katz M.J, Massie M.L and Culler D.E., Wide area cluster monitoring with Ganglia, in Proc. of IEEE International Conference on

Cluster Computing, pp.289 – 298, Hong Kong, December 2003.

[Samani et. al., 2012] Samani A., Ghenniwa H., Samarabandu J. "Risk-Based Modelling For Managing Privacy", IEEE Canadian Conference in Electrical and Computer Engineering (CCECE2012), pp. 1-5, May 2012.

[Sandholm et al., 2005] Sandholm, T., Suri, S., Gilpin, A. and Levine, D. CABOB: A Fast Optimal Algorithm for Winner Determination in Combinatorial Auctions. Management Science, 51, 3, 2005, 374-390.

[Schopf, 2001] Schopf J., (July 2001). "Ten Actions When Super Scheduling", document of Scheduling Working Group, Global Grid Forum, http://www.ggf.org/documents/GFD.4.pdf

[Schwartz and Solove, 2011] Schwartz P. M., Solove D.J., "The PII Problem: Privacy and A new Concept of Personally Identifiable Information", New York University Law Review, Vol. 86, 2011.

[Shan et al, 2004] Shan H., Oliker L., Biswas R., and Smith W., (December 2004). "Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration". Proceedings of ADCOM2004: International Conference on Advanced Computing and Communication, Ahmedabad Gujarat, India

[Shen et al., 1994] Shen, C., Pao, Y., and Yip, P., (1994). Scheduling multiple job problems with guided evolutionary simulated annealing approach, In Proceedings of the First IEEE Conference on Evolutionary Computation, pp 702-706.

[Shneidman and Parkes, 2003] Jeff Shneidman and David C. Parkes. Rationality and selfinterest in peer to peer networks. In 2nd Int. Workshop on Peerto-Peer Systems (IPTPS'03), 2003.

[Silva et al. 2003] Silva D. P., Cirne W. and Brasileiro F. V., (August 2003). Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids, in Proc of Euro-Par 2003, pp.169-180, Klagenfurt, Austria.

[Singh and Bawa, 2007] Singh, S. and Bawa, S. "Privacy, Trust and Policy based Authorization Framework for Services in Distributed Environments". International Journal of Computer Science 2(2):85-92, 2007.

[Solove, 2008] Solove D. J., “Understanding Privacy”, Harvard University Press, 2008.

[Stewart, 2000] Stewart, John E. (2000): Evolution’s Arrow: The direction of evolution and the future of humanity (Chapman Press, Rivett, Canberra, Australia)

[Subramani et al. 2002] Subramani V., Kettimuthu R., Srinivasan S. and Sadayappan P., (July 2002). Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests, in Proc. of 11th IEEE Symposium on High Performance Distributed Computing (HPDC 2002), pp.359- 366, Edinburgh, Scotland.

[Such et. al., 2012] Such J.M., Espinosa A., Garcia-Fornes A., “A survey of Privacy in Multi-agent Systems”, The knowledge Engineering Review, Vol. 00:0 pp:1-31, Cambridge University Press 2012.

[Sweeny, 2002] Sweeny L., “K-Anonymity: A Model for Protecting Privacy”, International Journal of Uncertainty, Fuzziness and Knowledge-based Systems - IJUFKS , 2002.

[TGA, 2013] The Globus Alliance (2013). http://www.globus.org.

[Tianchi et al 2005] Tianchi Ma and Rajkumar Buyya, (October 2005). Critical-Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids, in Proc. of the 17th International Symposium on Computer Architecture and High Performance Computing, Rio de Janeiro, Brazil.

[Topcuoglu et al. 2002] Topcuoglu H., Hariri S., Wu M.Y., (2002). Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 3, pp. 260 - 274.

[Tucker, 1998] Paul Tucker. Market mechanisms in a programmed system, 1998.

[Wellman, 1993] M. P. Wellman, “A market oriented programming environment and its application to distributed multicommodity flow problems,” Journal of Artificial Intelligence Research, Vol., 1, No. 1, pp.1- 23, 1993.

[Wellman et al., 2001] M. P. Wellman, E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, “Auction Protocols for Decentralized Scheduling”, Games and Economic

Behavior, 35(1-2), 271-303, 2001.

[Wieczorek et. al, 2005] Wieczorek M., Prodan R., and Fahringer T.. Scheduling of scientific workflows in the askalon grid enviornment. ACM SIGMOD Record, vol. 34, no. 3, pp. 56-62, 2005.

[Wright,2003] Wright D., (December 2003). "Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor", in Proceeding of Conference in Linux Cluster Computing (CLUSTER'03), pp.354-361, Hong Kong.

[Wolski et al., 1999] Wolski R., Spring N. T. and Hayes J., The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, in the Journal of Future Generation Computing Systems, Vol. 15, No. 5-6, pp. 757-768, January 1999.

[Yokoo, 2000] Makoto Yokoo. "Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems", Springer, 2000.

[Young et al., 2003] Young L., McGough S., Newhouse S., and Darlington J., (2003). "Scheduling Architecture and Algorithms within the ICENI Grid Middleware, in Proceeding of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '04, pp.240-245, Nevada, USA.

[Yu and Buyya, 2009] Yu J. and Buyya R. Gridbus Workflow Enactment Engine, Grid Computing: Infrastructure, Service, and Applications, L. Wang et al. (eds.). CRC Press, USA, 2009.

[Zhu, 2003] Zhu Y., A Survey on Grid Scheduling Systems, Department of Computer science, Hong Kong University of science and Technology, 2003.

Curriculum Vitae

Name: Raafat Aburukba

Post-secondary Education and Degrees:

The University of Western Ontario
London, Ontario, Canada
B.Sc. Computer Science with Software Engineering Specialization
1998-2002

The University of Western Ontario
London, Ontario, Canada
M.E.Sc. Software Engineering
2003-2005

The University of Western Ontario
London, Ontario, Canada
Ph.D. Software Engineering
2005-2013

Honors and Awards:

Natural Sciences and Engineering Research Council of Canada
2006-2009

Communications and Information Technology Ontario
2004-2005

Western Engineering Graduate Research Scholarship
2003-2009

Related Work Experience

Teaching Assistant
Electrical and Computer Engineering Department
The University of Western Ontario, London Ontario
2003-2009

Research Assistant
Cooperative Distributed Systems Engineering
The University of Western Ontario, London Ontario
2003-2013

Research Assistant
EK3 Technologies Inc., London Ontario
2003-2012

Publications:

1. Raafat Aburukba, Hamada Ghenniwa, Weiming Shen. "Bidding Specification Language and Winner Determination for Grid Computing Scheduling". *Proceedings of IEEE Computer Supported Cooperative Work in Design 2013, Accepted.*
2. Raafat Aburukba, Hamada Ghenniwa, Weiming Shen. "Economic-Based Modeling for Resource Scheduling in Grid Computing". *Proceedings of IEEE CSCWD 2012, pp.583-590.*
3. Raafat Aburukba, AbdulMutalib Masaud-Wahaishi, Hamada Ghenniwa, Weiming Shen. "Privacy-based computation model in e-Business", *International Journal of Production Research, Volume 47, Number 17, 2009, 4885-4906(22).*
4. Raafat Aburukba, Hamada Ghenniwa and Weiming Shen, "A Distributed Multi-Agent Approach for Collaborative Agile Manufacturing Scheduling", *International Journal of Agile Manufacturing, Vol. 10, Issue 1, 2007, 103-114*
5. Raafat Aburukba, Hamada Ghenniwa, and Weiming Shen. "Agent-Based Dynamic Scheduling Approach for Collaborative Manufacturing", *Proceedings of IEEE CSCWD 2007, Melbourne, Australia , April 26-28, 2007 , pp. 445-451.*
6. Raafat Aburukba, Hamada Ghenniwa, and Weiming Shen, "Agent-Based Approach for Dynamic Scheduling in Content-Based Networks", *Proceedings of the IEEE International Conference on e-Business Engineering 2006, pp. 425-432.*
7. Raafat Aburukba, Hamada Ghenniwa, and Weiming Shen. "Agent-Based Intelligent Media Distribution in Advertisement", *Information Technology for Balanced Manufacturing Systems, IFIP Series, Vol. 220, pp. 203-212, Springer, 2006.*
8. Raafat Aburukba, "An Agent-Oriented Approach for Dynamic Scheduling in Partially Overlapping Systems", *The University of Western Ontario*, Master of Engineering Science Dissertation, 2005.