CHAPTER 2

A SURVEY ON GRID SCHEDULING

2.1 OVERVIEW OF GRID SCHEDULING

Scheduling has been widely applied in the distributed systems before grid system emerges. However, grid concept has introduced specific requirements that make grid scheduling different from the conventional distributed system scheduling. A grid environment spans multiple administrative domains each with its own schedulers and management policies. Hence the grid scheduling requires additional mechanisms to find appropriate resources interacting with different local and remote policies and managing execution of jobs on those diverse administrative domains Belen Bonilla et al (2010).

In the cluster or decentralized scheduling which can be classified under conventional distributed system scheduling, the main goal of the users is to minimize the total execution time, i.e. makespan, for a set of jobs; however, in the grid environment, users may have various goals in addition to minimizing the makespan. More specifically, they may require a specific operating system, they may have data and memory requirements or they may pay attention to deadline rather minimizing the makespan. Consequently, a system framework, i.e. complicated software architecture, is needed to handle scheduling in the grid systems since local schedulers are far away to realize all those needs single-handedly.

2.1.1 Issues in Grid Scheduling

Scheduling in the grid system is a complex task due to various factors such as geographical distances, site autonomy, domain size and dynamic characteristics of resources. In this section, all these issues are explained including their possible solutions. The grid resources may be connected with relatively slower WANs in which transferring large input and output files can cause network latency. If this is the case for a particular grid environment, a virtual organization which is part of the scheduling framework, is adopted.

The conventional scheduling systems which refer to the local schedulers in the grid environment have a complete control over the resources. However, in the grid system placing a central control for these resources is rather difficult, since resources are distributed across autonomous domains. Therefore, a grid scheduling system should conserve site autonomy allowing local policies to run as a part of the grid scheduling framework. A decentralized scheduling mechanism is required not only as the resources are dispersed across different domains but also as there exist hundreds of sites, thousands of users and resources in a typical grid environment.

Another problem that grid schedulers should cope up with, is the dynamic characteristics of the grid resources. The availabilities and capabilities of resources vary rapidly in the grid environment. As a remedy, grid schedulers should choose online strategies instead of offline ones which assume constant availability of resources.

2.1.2 A Generic Grid Scheduling Process

In general, scheduling is performed in four phases in a grid environment. In the first phase, the user submits the jobs and QoS parameters. Then grid scheduler performs the resource discovery operation to find the resources which fit to the request. In the second phase, dynamic information (e.g. load on a resource) of these resources are gathered to specify if they are available to execute jobs within the timeframe. In the third phase, the scheduling strategy is applied to choose the best mapping, i.e. assigning jobs to the resources in the most appropriate way as mentioned by Schopf and Ramin Yahyapour (2002). In the final phase which is optional, execution of the jobs are monitored to make further improvements or to recover from failure.

2.1.3 Grid Scheduling Models

The structure of the scheduler affects the performance and scalability of the resource management system. The current scheduling systems can be categorized as centralized, decentralized, and hierarchical models.

In the centralized model, all jobs are submitted to a central scheduler that is in-charge of scheduling them on the appropriate resources. A central scheme may lead to a single point of failure in the distributed systems which yield poor scalability in the grid environment where the quality of resources and users can be extremely high. Moreover, this scheme cannot ensure site autonomy, which is a major requirement for grid computing.

The decentralized model addresses several significant problems such as fault-tolerance, scalability and site autonomy. However, the decentralized schedulers have to coordinate with each other using some complex protocols that will affect overall scalability of the system as quoted by Christian Grimme et al (2008).

In the hierarchical model, the schedulers are organized in a manner in which higher level schedulers control a larger set of resources and lower level schedulers control a small set of resources. This model is a combination of the centralized and decentralized models.

2.2 CONVENTIONAL GRID SCHEDULING STRATEGIES

The problem of finding an optimal scheduling in high computing distributed domains has been shown to be NP-complete as stated by (Fernandez-Baca, 1989). Therefore, heuristics have been developed to get near optimal scheduling solutions. The existing scheduling heuristics can be grouped into two classes namely Online (dynamic) and Offline mode (static) heuristics as described by Vazquez Poletti et al (2008).

Online mode heuristics assigns the jobs to the resources as soon as they arrive at the scheduler; however, in the offline mode heuristics jobs are collected into a set that is processed at predetermined intervals for mapping. Also, online mode heuristics map a job to a resource only once; however, offline mode heuristics may remap a job at each evaluation interval until it starts execution as mentioned by Vazquez-Poletti et al (2008).

Offline mode heuristics theoretically makes better decisions than online mode heuristics since they have the chance to deal with a group of jobs rather than a single one; however, this scheme has some drawbacks. As it is mentioned in section 2.1.1, the availabilities and capabilities of the resources vary rapidly in a grid environment. Thus, the dynamic information about resources which considers an offline heuristics can become quickly out of date. Moreover, resources may go offline, and in the next interval the heuristic has to reprocess jobs that have been already mapped to the previously online resources. It is apparent that this condition will dissatisfy the job owners who pay attention to makespan minimization. Actually, Online and Offline heuristics should be tested before deciding to place into a scheduling framework, since grid domains may present differences and may have different characteristics. Both online and offline mode heuristics assume that execution time of the jobs are known as a priority. The methods for estimating execution times based on task profiling and analytical benchmarking are discussed by Maheswaran et al (2009).

The various heuristics in the literature of distributed system scheduling have been proposed to be used in the computational grids, yet this section present the important ones briefly:

Online Mode Heuristics

- **FCFS**: First Come First Served strategy arbitrarily assigns jobs to the resources according to the arrival order.

- **OLB**: Opportunistic Load Balancing is a simple strategy gap that assigns each job in arbitrary order to the next available resource without regarding to the expected execution time on that resource. The intuition behind OLB is to keep all resources as busy as possible.

- **MET**: Minimum Execution Time, assigns each job in arbitrary order to the resource that gives the best expected execution time for that job, regardless of the current load on that resource.

- **MCT**: Minimum Completion Time assigns each job to the resource that offers the minimum completion time considering its next available time. The idea behind MCT is simply to join the benefits of OLB and MET.

Offline Mode Heuristics

- **Min-min**: Min-min heuristic has two phases: In the first phase, the minimum completion times are calculated for all jobs, next the job with the minimum overall completion time is selected from all the jobs and assigned to the corresponding resource. This process repeats until all the jobs are assigned to a resource.
- **Max-min**: Max-min heuristic applies the first step as Min-min; however, it assigns the job with the maximum completion time to the corresponding resource in the second phase. The Max-min heuristic is beneficial in the condition where completion time for the jobs varies significantly Liang Hu et al (2011).
- **Sufferage**: The idea behind Sufferage heuristic is that a resource is allocated to the job that would 'suffer' most in terms of expected completion time if that particular resource is not allocated to it. The sufferage value of the job is defined as the difference between its second best and its best estimated completion times. After the sufferage values of the jobs have been calculated, they are assigned correspondingly to the resources that offer the best completion times.

In reference Braun et al (2009) a set of heuristics, OLB, MET, MCT, Min-min, Max-min, Duplex, has been implemented and compared by simulation studies for independent job scheduling, i.e. jobs with no inter-task data dependencies in the heterogeneous distributed computing systems. In this study, the performance of the heuristics is based on minimizing the execution of the metatask. As the outcome of the simulations, the Min-min heuristic outperforms the other techniques.

Maheswaran et al (2009) also compare heuristics in their study and state that the selection of the heuristics to use in distributed system scheduling, depends on parameters such as heterogeneity among jobs and resources, and the arrival rate of the jobs.

In addition to comparison of heuristics, some other works aim to develop new heuristics. In the reference of Casanova et al (2000), Sufferage heuristic is improved as XSufferage to be capable of scheduling parameter sweep applications with files and I/O requirements. In Srisan and Uthayopas (2002), k-windows scheduling heuristic is proposed which schedules independent tasks on the clusters regarding the parameters such as available CPU load, free memory and the number of remaining jobs. In Vazquez-Poletti et al (2008). The QoS guided Min-min heuristic is presented in which the conventional Min-min heuristics is enhanced taking QoS criteria of grid users into account.

Beside heuristics, mathematical models, such as divisible load theory as described by Robertazzi (2003), have also been applied to solve scheduling problems in the grid computing. In the reference of Hung and Robertazzi (2004), divisible load theory is proposed to solve problem of allocating and scheduling computing resources on a grid environment for excessive number of independent tasks from large number of users. The divisible load scheduling can be applied to the optimization of distributed computing problems where both communication and computation load can be divided arbitrarily among a number of processors and links.

The aim of the divisible load scheduling, which uses a linear mathematical model, is to reduce the total execution time of the jobs on the computational resources. However, the divisible load theory based studies have a restriction of arbitrary-sized divisible load assumption, which is not necessarily the case in the real world. In reality, jobs and data are usually measured as integer number of units, and the integer programming problem is NP-complete. Therefore, the theoretical optimization that divisible load theory presents may lose its validity in a real world problem.

In most of the conventional grid scheduling studies Mathijs den Burger et al (2011), especially in simulation based ones, some common assumptions can be noticed. These assumptions force us to discuss them in the context of grid concept. First, there are only a few studies that consider scheduling dependent jobs as mentioned by Zhao et al (2004), typically, jobs are assumed to be independent that no inter-task dependencies exist; however, a grid domain should support both types of jobs. Second, in general, a few QoS criteria are considered; in most cases reducing makespan is the only goal. And finally no site autonomy is allowed, that is local sites could not run their own scheduling policies. Still, there are also two well organized studies; Ranganathan and Foster (2003) who consider most of the aspects mentioned above. They allow site autonomy by dividing scheduling into two phases namely external and internal scheduling. The external schedulers use site selection strategies to dispatch jobs and internal schedulers organize those jobs over local resources using heuristics.

Moreover, it allows priority to local policies and considers dynamic data replication as part of the scheduling problem, and considers dependent tasks and allows users to specify deadline for their jobs.

In addition to the theoretical and simulation works, a number of application level scheduling systems have been developed for grid computing, and they are explained and compared in several studies as discussed by Krauter et al (2002). The application level scheduling system provides a methodology, application software, and software settings for adaptively scheduling and deploying applications in heterogeneous, multi-user grid domains. AppLeS employs decentralized scheduling and uses services of Network Weather Service (NWS) to observe changes in performance of resources dynamically (Buyya 2002). InAppLeS, applications are embedded into agents that perform resource scheduling based on static and dynamic resource information provided by NWS.

Condor is a resource management system primarily designed for utilizing idle resources on a network. The condor employs a centralized scheduling scheme in which the user submits jobs to a central scheduler which is responsible for finding the appropriate resources for the execution of the jobs. The scheduler can transfer a running job among machines until it is completed. Each machine in the Condor system advertises its resources and reports their availabilities to the central scheduler. Condor-G, the complementary tool for Condor, merges the Inter-site resource management protocols of the Globus and intra-site resource and job managing schemes of Condor to let users exploit multi-site resources as though they all exist in a single site.

2.3 MARKET-BASED GRID SCHEDULING STRATEGIES

2.3.1 The Economic Problem

The economic problem exists only when the resources are scarce and the participants, namely consumers and producers, maximize their utility by deciding among needs that cannot be concurrently satisfied and lead to different utility levels. If the resources are reachable, whenever required, and their utilizations result the same in value, there would be no problem of allocating the resources for utility maximization, or in other words there would be no economic problem.

Considering the statement above, it is asserted that economic problem exists in the computational grids. Although there are number of resources are available with full potential, resources appear scarce due to heavy loads of the users, and allocation of the resources leads to different utility levels in the users' point of view since grid resources are heterogeneous. Therefore, scheduling in grid computing is very much concerned and related to the economics that is the study of how limited resources, goods, and services are allocated among competing users.

2.3.2 Economic Models for Grid Scheduling

In the real-world markets, various economic models are used for setting the price of a commodity or a service, based on the supply-demand and their value to the users as described by Buyya et al (2002). Some of these models are convenient to design a real grid economic system or solving the resource allocation problems.

Commodity Market Model

In a commodity market model, resource owners competitively set their service price and charge consumers according to the amount of resource they consume. The pricing policy of the commodity market model can be flat or variable depending on the supply and demand of the resources. In the flat price policy, price of a resource is fixed for a certain period and it remains the same regardless of the demand or service quality, whereas in a variable pricing policy, prices change frequently according to the supply and demand variations. As one of the main principles of economic theory, the increase of demand or the decrease of supply, increases the prices, and the decrease of demand or the increase of supply, lowers the prices until there exists an equilibrium between supply and demand.

Bargaining Model

In the bargaining model, consumers bargain with resource owners for lower service prices and longer usage periods. Both consumers and resource owners have their own utilization functions and they bargain with each other until they agree on a price or one of the sides is not willing to bargain any further. As a drawback, if the consumers start negotiation for getting relatively low prices then it may take considerable amount of time to reach an agreement causing waste of time.

Tender/Contract-Net Model

Tender/Contract-net model is based on the contracting mechanism used by businesses to manage the exchange of goods and services. The trading protocol of this model is simply as follows: First consumers announce their requirements such as memory, architecture and deadline whenever they require to commerce. In the second step, the interested resource owners evaluate the announcements and reply with their bids and finally, the announcement owner evaluates bids and start trading with the resource owner of the most appropriate bid. As a disadvantage, a less capable resource may be allocated if the capable resource is engaged in some other task and the consumers are not supported to inform the participating contractors that an agreement has been already made.

Auction Model

Auction by definition is the process of selling a property to the highest bidder. This model provides a one-to-many negotiation scheme between resource providers and consumers. The typical steps involved in an auction process are as follows: A resource owner announces his services and invites bids in the first step. Then, in the second step, consumers offer their bids. This step continues until no consumer is willing to bid a higher price or the resource owner ends if the minimum price value is not met with. The auctions can be carried out as open or closed depending on whether the consumers can see each other's offers or not. Additionally, consumers may raise the offered bid and auctioneers may update the offered sale price.

Community Model

In a community model, market players share their resources among themselves to form a cooperative computing environment. This model is suitable when those players have to be both producers and consumers. The trading can be realized by credits that are given proportionately in response to the amount shared by the resource owner.

Monopoly/Oligopoly

Monopoly is a market in which there is only one seller of a product or service. The consumers cannot affect the prices of the services and they have to buy the service at the given price. This may be the case in a grid environment if there is a single owner of a particular service. In addition, market situation can be an oligopoly if a small number of producers control the market and set the prices.

2.3.3 Market based Scheduling Studies in Conventional Distributed Systems

Adapting market-based approaches to scheduling problems in distributed computing system, is not a new idea; a number of systems had been developed before grid system was evolved. All those early studies make use of economics to obtain effective resource allocation solutions. Some of the important systems are explained in this section.

The Enterprise task scheduler utilizes the concepts of contract-net model to exploit idle resources in the distributed network. The scheduling procedure in the Enterprise system simply proceeds as follows: A client publicizes a task indicating relative priorities, description of required resources and information for machines to estimate the execution time. Subsequently, idle machines offer bids to the requests specifying the estimated execution time of the tasks. Finally, the machine that provides the shortest execution time is chosen by the client in order to process the task.

Actually, the enterprise system does not utilize a real market approach; it uses artificial priorities instead of a price mechanism which, in fact, can eliminate such a need. Hence, clients are unable to make decisions between fast and slow resources to submit tasks. Moreover, machines have no incentive to compete with each other in order to obtain a job by giving the best offer, since any utility maximizing behavior has not been defined for the machines.

The Walras system is an application of general equilibrium theory to management of distributed systems. The atonement process is implemented to attain the equilibrium price for the resources. In the atonement process, there is a central auctioneer that receives utility functions of the consumers and calculates the equilibrium prices using the Smale's theorem. Besides, for each resource it initiates a distinct auction, and after the calculation, the resulting prices are sent to all consumers. The consumers can respond to the new prices by adjusting their utility functions. This process is repeated until the prices reach a steady state.

Spawn is a scheduling system designed as a market economy that is composed of interactive consumers (buyers of CPU time) and resource owners, to run in the distributed computing environment. As an economic model, Vickrey auction is used to handle the sales. In addition to independent tasks, Spawn also supports concurrent tasks which are assumed to be tree-based. Each node in the tree which refers to a subtask, associated with a manager and has a funding rate shared by the root node. The managers of the nodes participate in an auction independently from each other. After the trading is over, they inform the root manager and deliver the results.

2.3.4 Market Based Scheduling Studies in Grid Systems

Assuming that the scheduling problem is decentralized in the grid system, market approaches can offer several advantages as stated by Wellman et al (2011). First, markets are naturally decentralized, and self-interested agents may yield global optimum solutions by focusing only on their own welfare. Second, communication is limited to the exchange of bids and prices among agents, and a pricing mechanism has the potential to differenciate resources more dynamically compared to a priority mechanism. Considering these advantages, several market-like systems (not real economies) have been proposed to attain optimal resource allocation solutions in the computational grid as in the conventional distributed systems.

A Contract-net based model is proposed for scheduling parallel jobs in the grid environments. In this work, simulations have been carried out to compare the proposed market-based method with the conventional First Come First Serve-Backfilling heuristic. The results reveal that their method outperforms the conventional one in terms of average-weighted-response-time and average-weighted-wait-time metrics. In Chien et al (2005), the continuous double auction protocol is used for the same purpose.

The continuous double auction outperforms the conventional FCFS and MCT heuristics in terms of job completion ratios and the protocol outperforms the conventional Round Robin heuristic in terms of weighted-completion-time metric as described by Chien et al (2005). In addition to these studies that compare market-like approaches with the conventional ones, in Gomoluch and Schroeder (2004) the atonement process is compared with the Vickrey auction in terms of grid wide price stability, market equilibrium, application efficiency and resource efficiency. The simulation results point that the atonement process is more successful than the auction model in controlling grid resources in terms of these criteria.

Market mechanisms can also be considered to build real computational grid economy to prompt individuals or companies share their resources and let the user benefit from the extensive resources in return for a fee.

There would be, naturally, two main players in a real computational grid market: Providers (resource owners) and consumers (resource users). Resource owners would like to maximize their utilities, i.e. make profits or recover their costs, by charging the users for accessing or using their resources and users would like to solve their problems by using the resources that satisfy their utilities i.e. budget, deadline and other QoS criteria. Moreover, the resource owners would like to offer a competitive service access price in order to attract users, and users would have the right to select the resources that best satisfy their needs. Thus, in the computational grid economy, a user would be in competition with other users and a resource owner with other resource owners as in all other real markets.

User applications may have various resource demands depending on computations performed and methods used in solving the problems. Some applications can be CPU intensive while others can be data intensive or a mixture of the two. Therefore, in economy grids, commodities can be a variety of resources such as CPU slots, memory, storage, and software and network usage. To charge the users for using these resources accounting services and payment mechanisms have to be deployed into a computational grid economy. He and Ioerger (2004) address this subject proposing accounting infrastructures and automatic payment mechanisms for computational grids.

Unfortunately, market-based real grid economy systems have not been commercialized yet; however, there are ongoing projects at the academic level. Among them Nimrod/G systems is presented as below

Nimrod/G is a market-based grid resource management and scheduling system, designed to be established in the real grid environments. It employs the Globus toolkit's services for dynamic resource discovery and shipping jobs over the grid, and employs Grid Architecture for Computational Economy framework (Buyya 2002) for the trading services.

There are five major components in the Nimrod/G architecture as shown in Figure 2.1. The Client or User System component operates as a user interface to control and manage jobs; users can specify time and cost constraints to affect the behavior of the scheduler in the resource selection phase, and they can monitor the status of the jobs. Parametric Engine is the central component that manages the entire application, i.e. set of jobs. Parameterization of an application, creation of jobs, maintenance of job status, and interacting with clients, schedule advisor, and dispatcher are the main responsibilities of this component. Moreover, it records the states of the entire experiment to restart it in case of failure. The Scheduler component is responsible for resource discovery, resources election and job assignment. The resource heuristics employed in the scheduler interact with a grid-information service directory, identify the list of available and appropriate machines, and keep track of resource status information. In addition, heuristics are responsible for selecting resources that meet the deadline and cost constraints. The Dispatcher component starts the Job Wrapper component that will initiate the execution of a job on the selected resource.

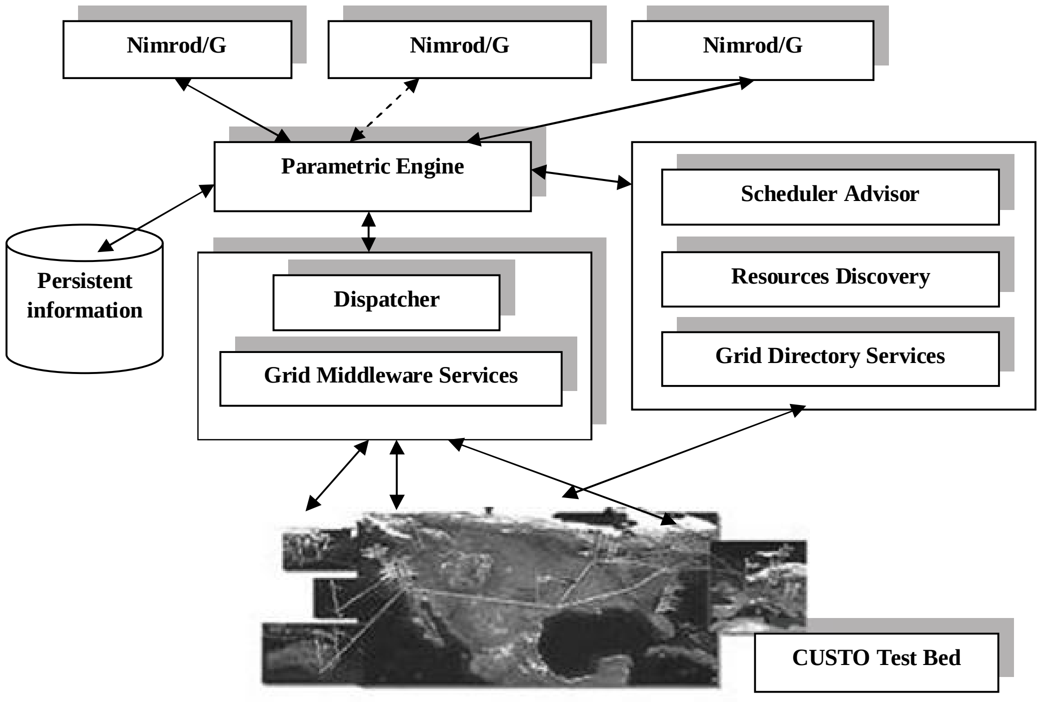

Figure 2.1 Nimrod/GArchitecture

The Compute Power Market (CPM) is another market-based resource management and job scheduling system designed by Buyya et al (2002) for grid computing. It transforms the distributed computing environment into a computational market in which users can solve problems by renting computational power.

The CPM is basically composed of markets, resource consumers, resource providers and their interactions. It provides various economic models such as commodity market model, contract-net/tender, and auction for resource pricing and scheduling.

The architectural components of the CPM are illustrated in Figure 2.2. Market is the point of interaction for both consumers and producers. Consumers and producers have to register with market to rent or sell computational power. A market basically provides repository of information on providers, agents for consumers and providers, mechanisms for updating the information and interaction with other markets.

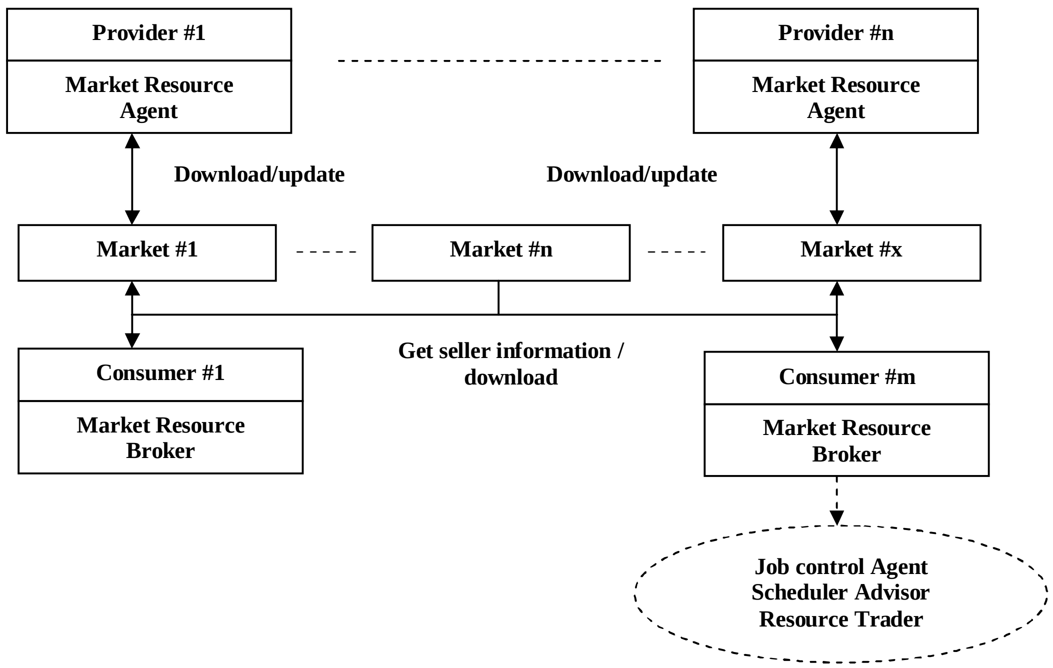

Figure 2.2 Components of CPM Architecture

Resource providers get a Market Resource Agent (MRA) from the market that is responsible for updating the market with the most recent information about their resources and accepting, deploying and launching the job. The MRA works on a push-pull mechanism with the pull unit extracting information such as available memory and number of processes, whereas the push unit sends this data into the market through communication unit. Correspondingly, resource consumers get a Market Resource Broker (MRB) that is responsible for locating the proper provider based on the information supplied by the market. The MRB comprises of several sub-components such as Job Control Agent, Market Explorer, Resource Trader, and Scheduler.

Job Control agent is responsible for guiding a job through the system by interacting with other components of the broker. An economy computation in the CPM grid is managed by schedule advisor, trade manager and trade server. The schedule advisor uses market explorer for resource discovery, trade server for negotiating costs and scheduling algorithms for deciding on mappings between jobs and resources.

2.4 RESEARCH MOTIVATION

The computing environments have evolved from single-user environments through Parallel Processors, clusters of workstations and distributed systems to the grid computing systems. Every transition has been a revolution, allowing scientists and engineers to solve complex problems and sophisticated applications which were previously incapable of getting solved. However every transition has brought new challenges and problems in its wake, as well as the need for technical innovation. The evolution of computing systems has led to the current situation in which millions of machines are interconnected via the Internet with various hardware and software configurations, capabilities, connection topologies and access policies.

The formidable mix of hardware and software resources on the Internet has kindled researchers’ interest in investigating novel ways to exploit these abundant pools of resources in an economical manner, as well as in aggregating these distributed resources so as to benefit a single and multiple applications.

The grid computing is diverse and heterogeneous in nature, spanning multiple domains whose resources are not owned or managed by a single administrator. This presents grid resource management with many challenges such as site autonomy, heterogeneous substrate and policy extensibility. The Globus middleware toolkit addresses these issues by providing services to assist users in the utilization of grid resources. The users are still exposed to the complexities of grid middleware, however there is a substantial burden imposed on them in which they must have extensive knowledge of the various grid middleware components in order to be able to utilize grid resources. Such knowledge ranges from querying information providers, selecting suitable resources for the user's job, forming the appropriate Job submission, submitting the user's job to the resources and initiating job execution.

A central component in the grid computing is resource brokering, which insulates the user from the complexities of grid middleware by performing the task of mapping the user's job requirements to resources that can meet these requirements. It can include searching multiple administrative domains in order to find a single resource which is to execute the job, or schedule a single job to use multiple resources across one or more sites. The resource brokering can be classified into two categories namely System-Centric and User-Centric. The System-Centric broker allocates resources based on parameters that enhance system utilization and throughput. Conversely, the User-Centric broker such as Globus adheres to the user's requirements.

The key goal of grid computing is to deliver utility of computations denied by these requirements. The grid resource brokers are thus focused on providing user-centric services. The grid must also cater to the need of multiple users from different sites, who may potentially be interested in the same resource, without knowledge of existence or interests of one another.

To maximize the benefits of grid computing, it is essential to be able to discover the resources available on the grid and to map the job to the most suitable resource.

In the computational grids, the problem of resource discovery and matching the job to suitable resources is a real challenge because of the large number of resources, their heterogeneity, availability and the variety of resource attributes such as CPU load and disk space.

In reality, there are already hundreds of grids around the world, each one developed to serve a specific group of researchers or users. The grid computing dream began with talk of creating an all-powerful grid: one grid is composed of many smaller grids joined together, forming a global network of computers that can operate as one vast computational resource. Across the world, researchers and software engineers are working to bring the grid closer to achieving the dream.

This research focuses on the problem of the grid economy and how grids can communicate with each other and exchange resources in order to satisfy job requirements for having reduced the number of rejection jobs.

2.4.1 Problem Definition

From the literature survey in the domain of decentralized market like grid, it is observed that

- It is also observed that there is no technique available with which unit price of resource utilization can be decided and fixed up by the consumer himself according to the economic condition or critical condition of meeting deadline for utilizing the resource available in the grid environment.

- Finally it is observed that there is no technique available in the domain which sorts out of the jobs as confirmed jobs and unconfirmed jobs through prediction in order to utilize the resources effectively without getting trapped into the infinite loop of serving the unconfirmed jobs wasting valuable utilization of time of the resources.

This research work has been carried out to accomplish the aforementioned aspects in the decentralized market like grid domain. Thus a comprehensive model namely Integrated Job and Resource Management (IJRM) Scheduling is proposed to accomplish the following three aspects.

- Accomplishing the discovery process.
- Fixing up of unit price for utilization of resources in the market like grid environment
- Sorting out jobs as unconfirmed jobs and confirmed jobs through the predictive technique in order to utilize resources effectively without wasting valuable utilization time of the resources.

Thus this proposed model is very beneficial to both consumer and resource providers in the suitable job and resource allocation.