CHAPTER 1

INTRODUCTION

1.1 GENERAL

Grid computing is a form of distributed computing that involves coordinating and sharing, computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations. It is highly promising technology which is capable of solving complex computational problems of an organization in scientific, engineering, and commercial applications.

The grid environment supports seamless access to available resources. It provides complete transparency among the network protocols of different platforms and administrative boundaries. The grid is basically a middleware and service infrastructure which is used for establishing managing and evolving multi-organizational federation called Virtual Organizations (VO). It provides an autonomous, dynamic, and domain independent system as quoted by Foster et al (2010).

A VO can be thought as a domain. Since all resources are geographically distributed, the grid provides a scalable virtual organization in which different resources are shared among themselves. The VO persistently follows up certain common goals though there is no central administrator who controls the entire system and mutual trusting relationship. The users,

information providers, and service providers such as application, storage, and CPU cycle server are main elements of the VO. Actually, the VO can be thought as a network, which is not of distributed a real environment like an office. As shown in Figure 1.1, various resources can be shared by one or more VOs.

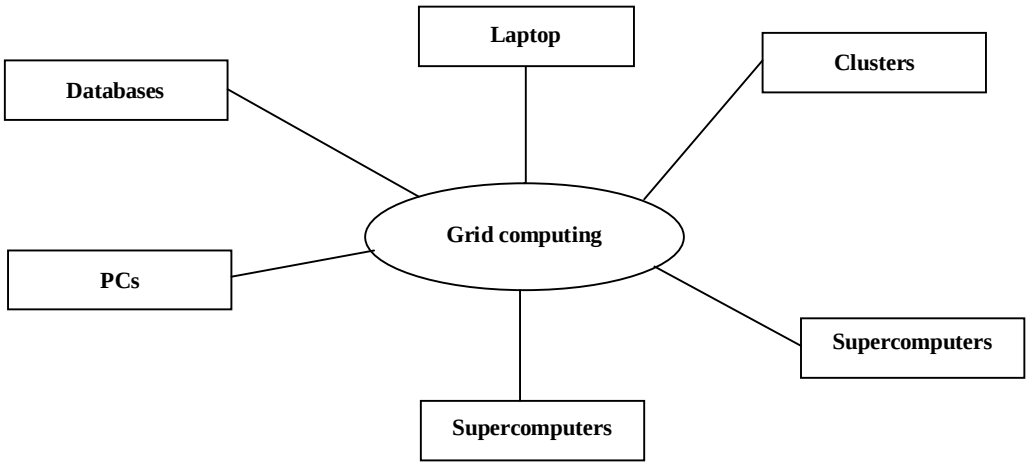


Figure 1.1 The Virtual Organization

1.2 BENEFITS OF GRID SYSTEMS

- The businesses make use of this technology for increasing their computational power, supporting data sharing and facilitating distributed workflow among different clients.
- A cooperative design and development can be provided by the system.
- Idle, available, computational and data storage resources can be exploited effectively.
- With the help of internet, the several scientific studies are carried out through distributed global collaboration among different researchers all over the world.

- It is used for making large data collections, maintaining large databases, performing large scale computing resources so as to have high performance in the distributed system.
- The grid is highly scalable as it can be easily expanded by plugging any resource to it.
- It provides transparency by hiding the complexity from the users while accessing the resources.
- It can be effectively used in the place of a super computer to perform critical tasks with considerable economy.

1.3 GRID ARCHITECTURE

The grid protocol architecture is composed of five layers namely Grid Fabric Layer, Grid Connectivity Layer, Grid Resource Layer, Grid Collective Layer and Grid Application Layer Foster et al (2010). It is a layered architecture like Internet protocol architecture as shown in Figure 1.2 and the function of each grid layer is briefly presented as follows.

1.3.1 Grid Fabric Layer

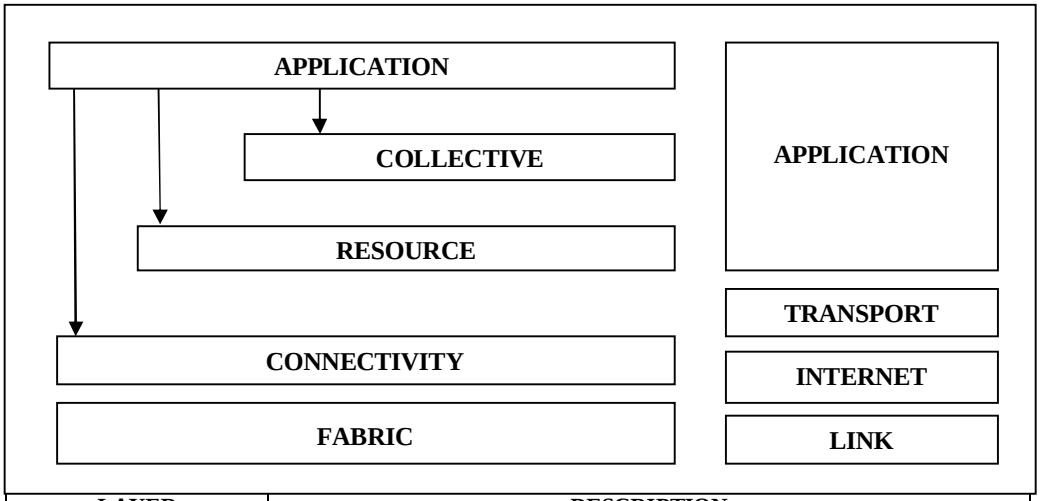
Fabric layer is composed of resource-specific and site-specific mechanisms. It has low-end and high-end computers including clusters, networks, scientific instruments, and also resource management mechanisms. Examples of these mechanisms include resource management, interfaces supporting, and advanced reservation which make it possible for higher-level services to aggregate resources effectively otherwise it would be impossible to achieve some network Quality of Service (QoS) in routers. The fabric layer also provides administrator sharing operations for resources at higher levels as quoted by Foster et al (2010).

1.3.2 Grid Connectivity Layer

It describes authentication and authorization protocols and core communication protocols for the grid-specific network transactions. These communication protocols can be used between the fabric layer resources to exchange data. The connectivity layer also provides some security mechanisms such as single sign on, delegation and user-based trust relationship.

1.3.3 Grid Resource Layer

The resource layer builds on connectivity layer and it defines the protocols for secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. It also offers remote process management, co-allocation of resources, storage access, information security, and QoS like resource reservation and trading (Zhang 2002).



LAYER	DESCRIPTION
Collective (application specific)	Distributed data archiver, check pointing, job management, failover and staging.
Collective(generic)	Resource discovery, resource brokering, system monitoring, community authorization and certificate revocation.
Resource	Access to computation access to data, access to information about system structure, state and performance.
Connectivity	Communication, service discovery, authentication, authorization and delegation.
Fabric	Storage systems, computers, networks, code repositories and catalogs.

Figure 1.2 Grid Architecture

1.3.4 Grid Collective Layer

The collective layer focuses on protocol and services which are related to collections of resources. The services such as directory services allow users to query for the availability of resources and current load. The Co-allocation, scheduling, and brokering services allow users to request resources for their tasks scheduling on suitable resources among different available resources. The monitoring and diagnostics services provide the monitoring of resources failure, Intrusion Detection, network failures, current resource load and overload. The data replication services replicate data to increase data access performance like minimizing response time, maximizing reliability and minimizing cost.

1.3.5 Grid Application Layer

This last layer is composed of the applications of users that are developed using grid-enabled languages like High Performance C++, and message passing systems like Mozilla public license applications enforced with Grid Services, Grid Fabric mechanisms and Application Toolkit components.

1.4 USAGE BASED GRID CATEGORIES

The grid systems can be grouped into seven categories according to their usage. These are as follow:

1.4.1 Traditional Grid

The traditional grid is a closed computer network that implies both its availability to only a few consumers and it is being maintained by a single

administrator. For example, the traditional grids are often built for a specific test e.g. National Aeronautics and Space Administration (NASA) Information Power Grid as mentioned by Amin et al (2003). It is used for official business and research that is only open to scientists and engineers working for NASA. The traditional grids are mostly homogenous. They offer maximum performance because of their single ownership. On the other hand, these systems have minimal flexibility as they are developed for specific goals as described by Claassens and Vander Weegen (2007).

1.4.2 Modern Grid

Contrary to the traditional grid, the modern grid is open, heterogeneous and flexible. However these good properties bring some problems such as secure negotiation, authentication, authorization and sharing that need to be solved. For example, the transmission speed capacity of Ethernet is increased from mega bytes to giga bytes. The ability of Ethernet to scale easily is a contributing factor to its continued use of modern grids.

1.4.3 Computational Grid

A computational grid consists of resources that are specifically designed for performing various computations in the grid environment. It has mostly high-performance server. A computational Grid is a system that provides higher aggregate computational power than any single personal machine. According to utilization of the computing power, the computational grids can be further divided into two subcategories namely Distributed Supercomputing Grid and High Throughput Grid. A Distributed Supercomputing grid utilizes the parallel execution of applications over multiple machines simultaneously to reduce the execution time. On the other

hand, the goal of the High Throughput grid is to increase the completion rate of a stream of jobs by utilizing available idle computing cycles as much as possible. The computational grid should not be thought of as a parallel computing because there is no single owner responsible for all systems. For example, if a researcher needs to make CPU calculation, he could occasionally borrow CPU time from a grid to much lower cost than borrow the time from a super computer. A grid could be created in all environments where end users have a computer with CPU and a memory.

1.4.4 Data Grid

The purpose of the data grid is to build the next generation computing infrastructure supporting intensive computation and analysis of shared large-scale databases across widely distributed scientific communities as proposed by Christopher Moretti et al (2010). In the data grid, consumers are not concerned with where these data are located and how they can be accessed. The data are stored in different locations and efficiently shared among several consumers. For example, In case of more than two universities working on a space research which requires a high volume of data, they can build a data grid among themselves to share and manage the data considering security issue. European Data Grid Project (European Union 2005) is an example for data grids.

1.4.5 Scavenging Grid

The desktop PCs are used to design such type of grid. In the scavenging grid, there is a central scheduler responsible for connecting the resources. When a desktop machine becomes idle, it reports its idle time to the scheduler. For example in Seti@Home Anderson Cobb and Korpela

(2002) project, when a desktop machine becomes idle, Seti@Home's client program enters the process and connects the machine to the grid. Then all resources work for the project. In the scavenging grid, the user does not need to set up any Operating System or middleware like Globus. In the scavenging grid, a desktop machine works for grid on a voluntary basis.

1.4.6 Storage Grid

A storage Grid attempts to aggregate the spare storage resources in the grid environment and provides users with transparent and secure storage services. For example, Network Attached Storage (NAS) and Storage Area Network (SAN) provide shared storage for multiple servers and multiple protocols. For example; Berkeley SAN provides more than 30 Terabytes of Premium and Enhanced storage to more than 20 clients and expect long-term count to be closer to 100 TB.

1.4.7 Peer to Peer Grid

It is based on Peer to Peer (P2P) and Grid technologies and it takes advantage of both. The P2P is a resource sharing mechanism available at the edge of the internet through adhoc overlay networks with symmetric communication Sujoy Mistry et al (2011). A P2P computer network is a network which depends on the computing power and bandwidth of the participants in the network. In the P2P, instead of creating a huge network of computers, like the Internet, one should be able to connect directly to the system which can provide your need. It prevents overhead considerably. Only computers that can be run under the same software (Chord, Pastry, Best Peer, Ares, and Emule) can be connected together. The P2P can be thought of as a variant of data grids because the aim of it is data exchange. The tasks are

distributed to grid nodes in a decentralized manner. Because of its ability and robustness, it is widely used in many cases. However, existing P2P networks do not guarantee that the shared data can be always located at the same place. In order to provide a persistent and stable network, the P2P grid is developed Iamnitchi et al (2002).

1.5 TYPES OF GRID TOPOLOGIES

The grids can be built in different sizes to meet the different requirements. A grid can be a group of few machines in a department or campus networks or a group of computers organized as a hierarchy. In general, there are three types of grid system available according to their topologies namely Intragrid, Extragrid and Intergrid as shown in Figure 1.3.

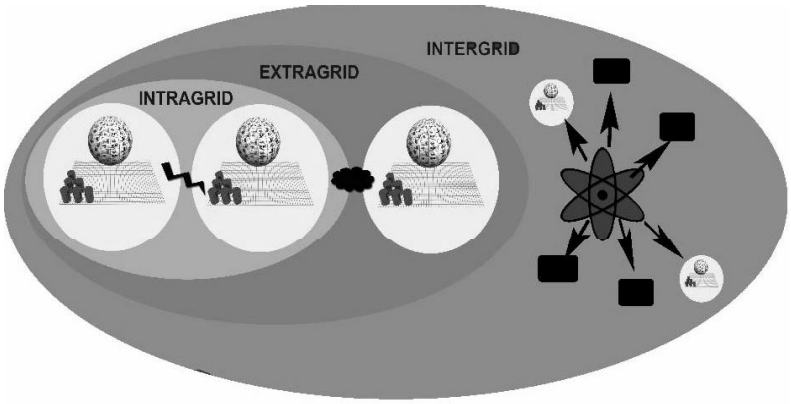


Figure 1.3 Types of Grid

As shown from Figure 1.4, the simplest type of grid is Intragrid. It is used to design a simple network for an organization with multiple departments which provides single security management supporting high bandwidth.

The Intragrid consists of several secure homogeneous systems which are operated on separate Operating System and hardware components connected through Local Area Network (LAN).

The Intragrid consist of NAS and SAN to provide shared storage for multiple servers and multiple protocols. For example, UC Berkeley Storage Area Network provides more than 30 Terabytes (TB) of Premium and enhanced storage to more than 20 clients and expect long-term count to be closer to 100 TB. The advantages of SAN technology include increased availability, enhanced performance, and better monitoring through centralized administration as stated by Berkeley et al (2005).

The Extragrid consists of several partners with more than one security partner which is operated on a same or different operating systems and hardware components connected through Virtual Private Network (VPN).

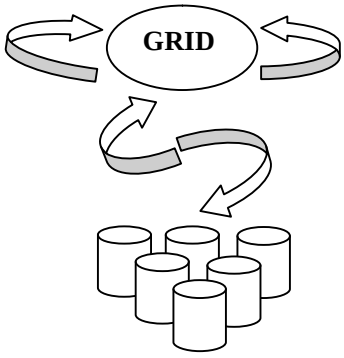


Figure 1.4 Intragrid

It is used to perform business to business transactions among trusted business partners in order to establish an amicable relationship. The Intergrid consists of many organizations with multiple partners which work

on global data putting different grids together and it can be primarily used by engineers in different firms, manufacturing, life science industries and financial industries (Figure 1.5).

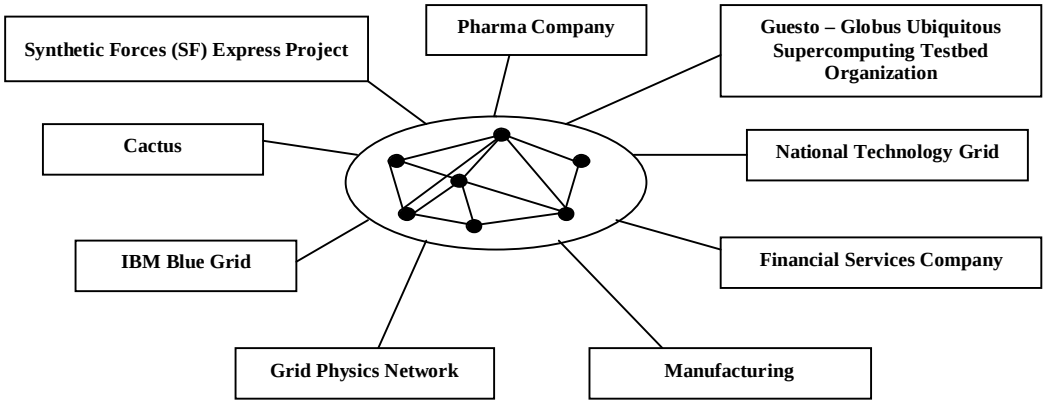


Figure 1.5 Intergrid

1.6 TYPES OF RESOURCES IN A GRID

The different kinds of resources such as computational storage, databases, communication links, software and licenses, special equipments, architectures and policies are available in the grid.

1.6.1 Computational Resources

The computational resources which can be varying in speed, architecture, software platform and connectivity are the most common and important resources in the grid. They enable CPU scavenging to utilize resources effectively. If any computer becomes idle, it will report its state to grid so that it can be joined to the grid and afterwards it becomes a part of resource of the grid. The computational grid aggregates the processing power from a distributed collection of systems. They provide the computational

power to process large scale jobs. The computational resources satisfy the business requirement for instant access to resources on demand. In the grid system, there are three major ways to utilize the computation resources:

- An existing parallel application can be run on the grid.
- Applications or tasks can be divided into separate parts and execute in parallel on different machines in the grid.
- Application is run many times on many different machines at the same time.

1.6.2 Data Storage Resource

It is the second most common resource in the grid. It can be a memory attached to the processor or it can be a secondary storage such as hard disk drive and type driver to increase capacity, performance, sharing and reliability of data. The data grid provides a scalable storage and access to the datasets. The replicated, catalogued, and even different datasets are stored in different locations to create an illusion of mass storage. Using unified file systems such as Andrew File System and Network File System with the storage on multiple machines, the capacity can be increased. These advanced file systems can duplicate sets of data. Meanwhile, an intelligent grid scheduler can help to choose suitable storage device to hold data depending on the usage job patterns. Moreover in the grid system, journaling can be implemented by the grid file system so that the data can be recovered in a more reliable way after the failures. Since data are shared and updated by lots of users, grid file system performs advanced synchronization mechanism to reduce contention between these users (Figure 1.6). Also data striping in

writing or reading successive records to/from different physical devices overlap the access for faster throughput as quoted by (Berstis 2002).

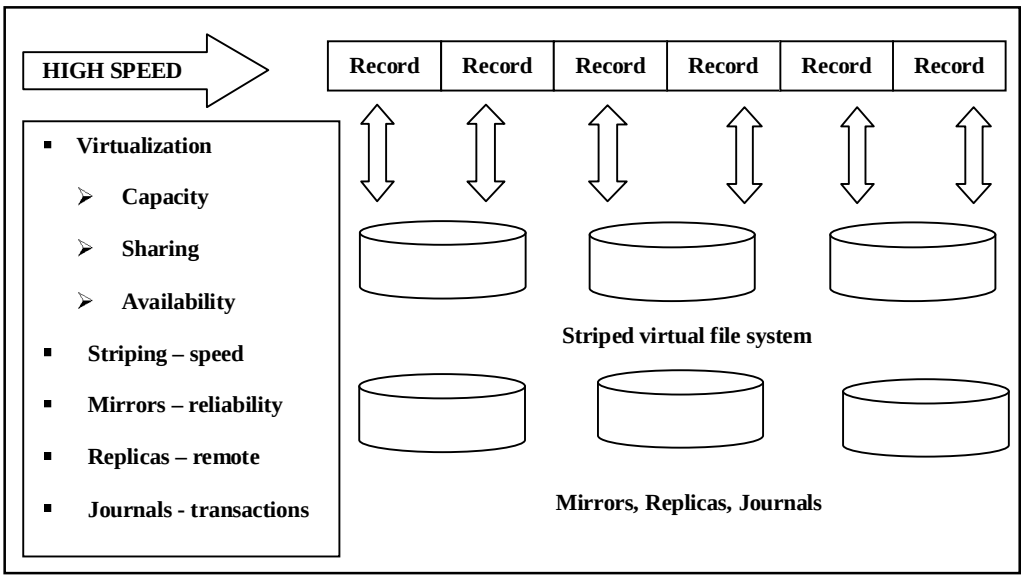


Figure 1.6 Data Grid

1.6.3 Communication

It becomes another resource in the grid when some jobs require a lot of data to be processed since bandwidth can be a critical resource that can limit utilization of the grid for such jobs. Sometimes to overcome potential network failures and huge data traffic, redundant communication paths such as VPNs are needed Haiying Shen et al (2012). In an Intergrid, if it is assumed that a search engine that should access the external Internet, is developed to provide connectivity among the grid machines, these connections do not want to share the same communications path, but they want to add the new total available bandwidth for accessing the Internet.

Software and licenses, special equipment, capacities, architecture and policies represent a different kind of resources. The Installation of very

expensive software on every grid machine increases the cost. To prevent this, this software is installed on some particular machines in which jobs require this software. Therefore this method can reduce the cost for an organization. On the other hand, some software licensing arrangements allow the software to be installed on all of the machines but may limit the number of software instances that can be simultaneously executed at any given time. This limitation is enforced by license management software.

Because of the heterogeneous and the dynamic nature of the grid, it has often different architectures, Operating Systems, devices, capacities, and policies. Each of these items represents a different kind of resource while the grid assigns jobs to machines since it can use this special equipment, capacities, architecture and policies as criteria. For example there are several types of software running on different architectures such as x86, SGI and Sun Ultra. Therefore the users must consider such characteristics ahead when assigning the jobs to machines in the grid. In general, the grid resources are classified as shown in Figure 1.7. A Grid resource can have resource ID, resource name, cost (price), and performance criteria.

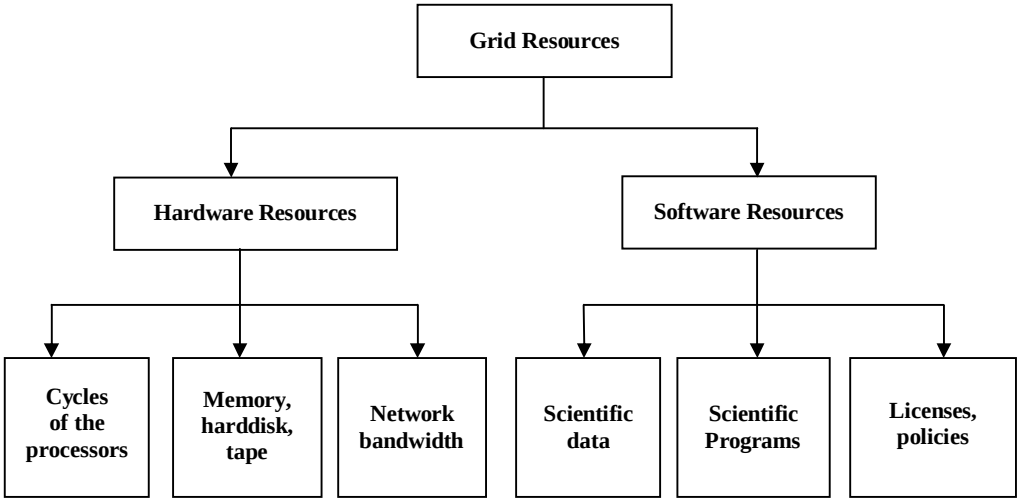


Figure 1.7 A Classification of Grid Resources

1.7 GRID SCHEDULING

The management of grid resources in a grid environment is clearly important. The overall aim is to obtain the efficient and effective scheduling of the applications that need to utilize the available resources in the distributed environment. The grid scheduling is essential to provide consistent and coordinated use of heterogeneous Grid. From a user's point of view, resource management and scheduling should be transparent.

Because of the heterogeneous and the dynamic nature of the grid, scheduling is significantly complicated due to the difference in performance goals by various grid applications (users) and grid resources. Most grid systems use grid scheduler responsible for resource discovery and available resources by selecting the most suitable resources that provide user's request, and assigning the task onto selected resources. A grid system responsible for executing a job does this in two ways namely simplest of grid system and advanced grid systems. In the simplest of grid systems, the user selects any machine or any resource suitable for running his job. On the other hand, in a more advanced grid systems, the job scheduler finds the most appropriate machine to run a given job as mentioned by (Berstis 2002).

There have been lots of scheduling algorithms introduced for homogeneous and dedicated resources like computer clusters. Since clusters have monotonic performance goal, high speed interconnection network, dedicated resources, homogeneity of resources and applications, these scheduling algorithms (e.g. First Come First Serve, Minimal-Requested-Job-First, Shortest Job First, Backfill and so on) have a complete control over all computing nodes and the overall information about the pending jobs. Thus, the scheduler can schedule jobs onto the cluster effectively and efficiently.

However these algorithms cannot be suitable and work well for current heterogeneous environments like Grid. Therefore, the grid must deal with large-scale heterogeneous resources across different management boundaries.

In such a dynamic distributed computing environment, resources availability varies dramatically. For example, while some resources may go offline due to some network and hardware failure, others may go online. The scheduling in the grid environment is significantly more complicated than the traditional scheduling systems for distributed environments. The grid scheduling systems must take both the diverse characteristics of grid applications and various grid resources into account. Therefore, scheduling becomes quite challenging in the grid.

The ordering and mapping are two deterministic characters in the grid scheduling. The ordering is applied when there is more than one task waiting for execution. It can be determined by which order the pending applications can be arranged. The ordering must be done according to applications' priority or deadline factors. This means that the access to resources is typically subject to individual access, priority, and deadline policies of the resource owners. On the other hand, mapping is the process of finding the most suitable resource and then assigning the applications to such resources. For the best scheduling, the performance potential should be estimated for each mapping. The scheduling in grid computing means that one or more users' tasks that can be submitted without knowing where the resources are or even who owns the resources. A Grid scheduler or scheduling algorithm has to guarantee the QoS of a job's execution. In this context, the following terminologies are used as stated by Zhu et al (2005)

- A **task** is a single indivisible unit to be scheduled by any grid scheduler.
- The **properties** of a task are parameters like length, deadline, priority and CPU requirements.
- An **application** (or **job**) is composed of more than two tasks and each task has sub-tasks that have an ability to decompose themselves further into atomic tasks
- A **resource** is something that is required to carry out an operation, for example; computational resource, network resource, software resource, data, database, instrument, code repository and storage space.
- A **node** (or **site**) is composed of one or more grid resources.

Each job consists of a number of tasks. While scheduling this task to resources, several challenges may be encountered as follow:

- Resource heterogeneity: How does the heterogeneity of resources affect the performance of a schedule?
- Site Autonomy: How do the site's scheduling policy, local priority, and security and management policy affect the quality of a schedule?
- Non-Dedicated Resources: Here contention is a major issue and it can exist both in the computational resources and network connections. Moreover, a resource may join multiple Grids simultaneously if it is non-dedicated.

- Application Diversity: In Grid, applications are from wide range of users each of whom has its own special requirements.

According to Casavant and Kuhl (2007), the scheduling problem consists of three main components namely consumers, resources and policies. The policy may specify the objectives that a scheduling system may satisfy. It may also specify, at the implementation level, the method of mapping jobs to resources. The policy chosen for implementing a scheduler affects both users and resource providers. A scheduler is designed to satisfy one or more of the following common objectives:

- Maximizing resource utilizations
- Maximizing system throughput (it is the amount of work that a computer can do in a given time period)
- Maximizing economic gains (for system/resource owner perspective)
- Minimizing the turn-around time for an application/job/meta task
- Minimizing the overall execution time of a job
- Balancing the load among the grid resources
- Minimizing the makespan is the total time that lasts until the last task in the job is done, Minimizing cost of the processors from the users/consumers perspective
- Minimizing the communication cost

- Meeting the users' QoS requirements from the user perspective
- Meeting the users' deadline and budget request from the user perspective. It is also possible to classify the scheduling according to their primary object Aggarwal et al (2005).

    1. Application Centric
        a. Makespan
        b. Economic Cost
    2. Resource Centric
        a. Resource Utilization
        b. Economic Profit

In the most general case, scheduling algorithms have to be adapted to the different optimization criteria that a user can specify for each particular job. Some of the most frequent optimization criteria for a user are as follow:

1. Optimizing performance without regard to the cost.
2. Optimizing cost without regard to the performance
3. Optimizing performance within a specific time constraint.
4. Optimizing cost within a specific time constraint.
5. Optimizing performance within a specific time and cost constraint.
6. Optimizing cost within a specific time and cost constraint.

1.8 TAXONOMY FOR GRID SCHEDULING ALGORITHMS

In general, Adil Yousif et al (2011) introduce a hierarchical taxonomy for scheduling algorithms and distributed computing systems. Since Grid scheduling is similar to this system, it can be thought as a subset of this taxonomy. The taxonomy is based on this approach as shown in Figure 1.8.

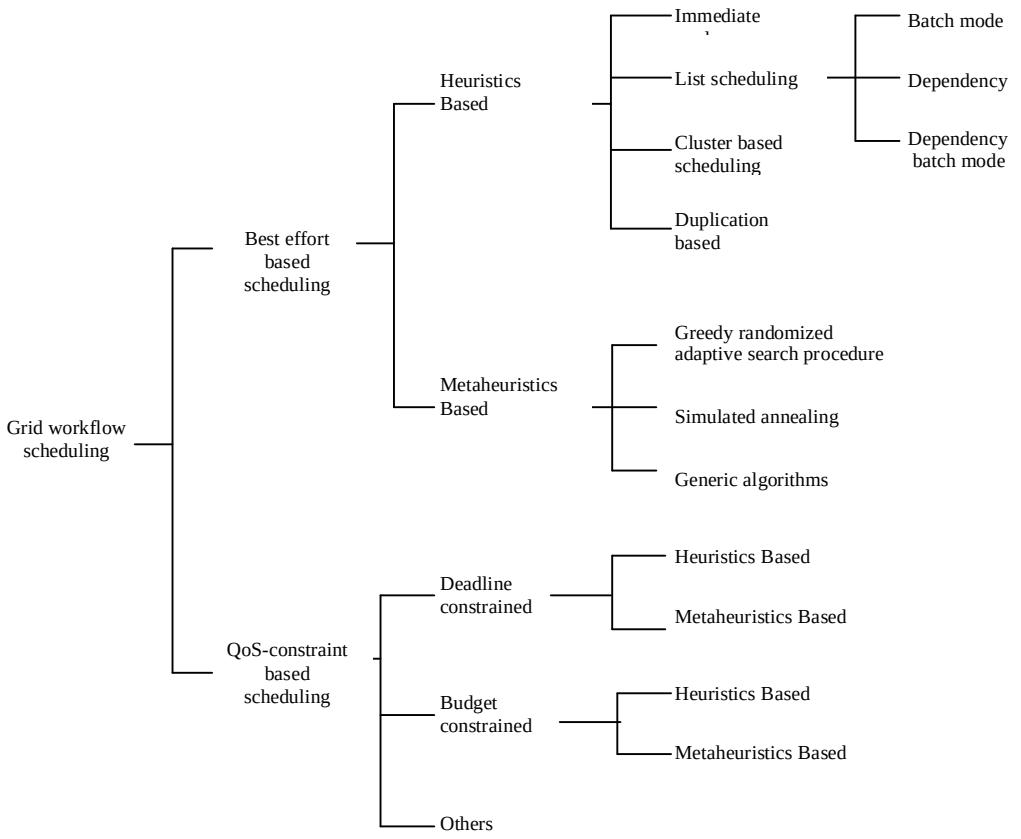


Figure 1.8 Taxonomy of Grid Scheduling

The global scheduling can be classified into two sub categories. One of them is decentralized scheduling model. In this model, there is no central leader responsible for scheduling. Therefore there is no disadvantage of having centralized scheduling model. The decentralized scheduling strategy is suitable for grid systems because resource owners can define their

scheduling policies that schedulers can enforce. However as the resource owners may not agree on a common policy goal for resource management, development of scheduling algorithms is difficult.

The association job allocation technique is the most suitable scheduling scheme for grid systems because this model permits the remote resource owners to enforce their own local policy on external users. This means that different schedulers may take independent decisions at each level of hierarchy (Zhang 2002).

The heuristic scheduling technique is suitable for dynamic structure of Grid because it dynamically evaluates application efficiency and its execution times. The number of resources is adapted with the help of this information and finally the applications are assigned to the resources Huedo et al (2004). According to the author in (Greenshields 2002), with adaptive scheduling technique, scheduling strategy and parameters can be replaced at runtime.

The studies have shown that multi-agent systems may have been used successfully to overcome a wide range of complex distributed scheduling problems, since multi-agent systems both have autonomous, distributed and dynamic nature, they are robust against faults. Therefore, complex, robust, and cost-effective next-generation scheduling systems can be built with Multi-agent systems. In Cao et al (2007), author claims that the foundation for the creation of grid scheduling systems that have capabilities of autonomy, heterogeneity, reliability, maintainability, flexibility, and robustness can be provided by the decentralized systems.

1.9 THESIS OUTLINE

In this section, the outline of the various chapters of the thesis is set out.

- Chapter 2 discusses the-state-of-art relevant to the work in this thesis. It elaborates the P2P decentralized coordination, allocation scheduling, optimization and economic factors regarding cost and time.
- Chapter 3 reviews and provides job resource allocation called Association based grid allocation algorithm.
- Chapter 4 presents the economic cost and time management scheduling called Demand and Auction based grid scheduling algorithm.
- Chapter 5 implements the optimization for job scheduling and maximizes resource utilization called Heuristic based grid optimization algorithm.
- Chapter 6 combines the coordination, allocation, scheduling and optimization and presents a comprehensive model namely called Integrated Job and Resource Management (IJRM) scheduling.
- Chapter 7 presents the conclusion drawn in this thesis work along with the possible future scope of research directions which can be shown in this research work.