1

Introduction

1.1 Grid Computing

Grid computing is an emerging computing paradigm which is inspired by the electrical power grid. Looking at the ease of use, pervasiveness and reliability of the electrical power grid, computer scientists too started exploring the design / development of an analogous infrastructure for wide-area parallel and distributed computing and data sharing. This resulted into the notion of computational power grid [1]. The motivation for computational Grids was initially driven by large-scale, resource (computational and data) intensive scientific applications that require more

resource than a single computer (PC, workstation, supercomputer, or cluster) could provide in a single administrative domain. Grid computing strives to aggregate diverse, heterogeneous, and geographically distributed and multiple domain spanning resources to provide a platform for transparent, secure, coordinated, and high-performance resource-sharing and problem solving [2]-[4].

Grids are becoming platforms for high-performance and distributed computing [3]. Grid computing can be thought of as an enhanced form of distributed computing. A grid benefits users by permitting them to access heterogeneous resources, such as machines, data, people and devices that are distributed geographically and organizationally. It benefits organizations by permitting them to offer unused resources on existing hardware and software. They allow users to execute compute intensive problems whose computational requirements cannot be satisfied by a single machine [4]. A computational grid environment behaves like a virtual organization consisting of distributed resources. A virtual organization is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place [4]. Grid computing has emerged as the new paradigm in distributed computing and has undergone significant developments over recent years.

Grid is type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost, user quality-of – self-service requirement. [2]

Foster, et.al. defined the grid problem as coordinated resource sharing and problem solving in dynamic multi-institutional, virtual organization [4]. The definition of grid conveys the following distinct dimensions of the grid:

"*Resource*" in a broad sense means data, computers, memory, scientific instruments, software, peripherals, network bandwidth etc.

"*Coordinated Sharing*" Resources together with their providers / consumers are clearly defined, and in that multiple resources may need to be organized in an integrated fashion to achieve various qualities of service. Achieving this coordination involves the establishment and enforcement of sharing agreements.

"*Dynamic Membership*" as participants may leave or join at any time.

1.1.1 Benefits for Grid Computing

The main objective of the grid computing is to create the illusion of a simple yet powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources [5]. Grid computing provides some of the potential benefits as listed below.

1.1.1.1 Full utilization of idle resources

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines are busy less than 5% of the time. In some organizations, even the server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage.

1.1.1.2 Parallel CPU capacity

The other attractive feature of the grid computing is massive parallel CPU capacity. This computing power is driving factor for evolution in industries like bio-medical field, financial modeling etc. along with scientific applications.

1.1.1.3 Virtual organizations / Resources

Grid computing enables the heterogeneous systems to collaborate by offering the important standards and hence creates the illusion of a large virtual computing system with a variety of virtual resources. The users of the grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger grid.

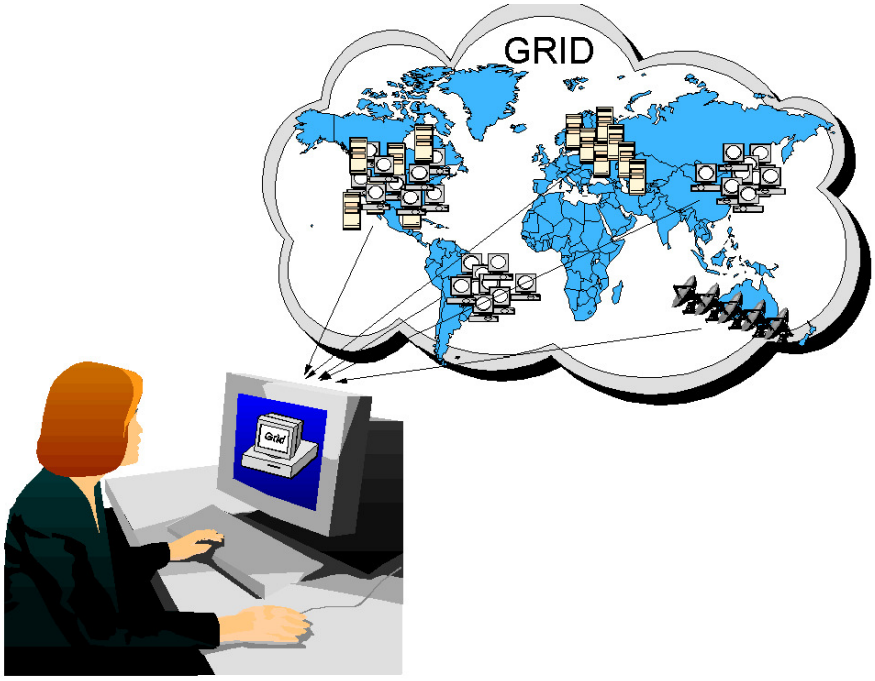

Figure 1.1 Virtual Organization (Simpler view) [5]

1.1.1.4 Load balancing

A grid unites a large number of resources contributed by individual machines into a greater total virtual resource. For grid enabled applications, the grid can offer a load balancing effect by scheduling grid jobs on machines with low utilization.

1.1.1.5 Reliability

Grid computing provides a software technology based approach to assure the reliable operation than using expensive hardware. Since the systems in the grid are

geographically dispersed so break down at one location leaves other parts of the grid unaffected. As soon as a failure is detected, Grid management software automatically resubmits the jobs to some other machines on the grid.

1.1.2 Types of Grid

A grid is a collection of machines, sometimes referred to as nodes, resources, members etc. They all contribute any combination of resources to the grid as a whole. Depending on the type of resource being shared a Grid can be broadly divided into two categories: Data Grid and Computational Grid.

1.1.2.1 Data Grid

The most common use of the grid is data storage. Data grid is an integrated view of data storage [6]. Every machine connected to the grid provides some quantity of storage for grid use. Storage can be primary or volatile memory or secondary memory using hard-disk or some other type of permanent storage media like magnetic disc or tape etc.

Secondary storage shared on the grid can be used in interesting ways to increase the capacity, performance and reliability of the data. Most of the grid systems use mountable networked file systems. Capacity can be increased by using the storage on multiple machines with a unifying file system. Any individual file or data base can span several storage devices and machines, eliminating maximum size restrictions often imposed by file systems shipped with operating systems. A unifying file system can also provide a single uniform name space for grid storage.

More advanced file systems on a grid can automatically duplicate sets of data, to provide redundancy for increased reliability and increased performance. Data striping can also be implemented by grid file systems. Similarly by implementing

journaling in a data grid, data can be recovered more reliably after certain kind of failures.

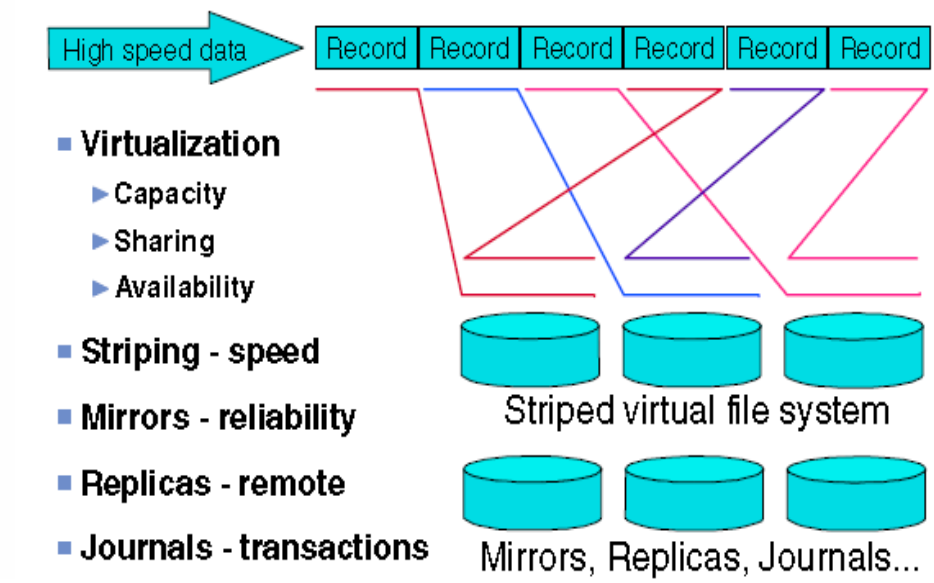

Figure 1.2: View of Data [5]

1.1.2.2 Computational Grid

Exploiting the computing cycles provided by processors is another use of Grid computing. Many scientific applications are compute intensive that require a lot of computing power to find out some optimal or sub-optimal solution. Computational grids aim to exploit the computing cycles of idle processors connected in the grid and provide a view of virtual supercomputing with number of processing elements working in parallel to provide the solution to a problem which is otherwise impossible to tackle with normal desktop computers. There are different ways to exploit the computing power of a grid resource.

- ✓ Run an existing application on an available machine on the grid rather than locally.
- ✓ To design an application to split its work so that different parts can be executed on different machines in parallel.

- ✓ To run an application that needs to be executed many times on different machines in the grid.

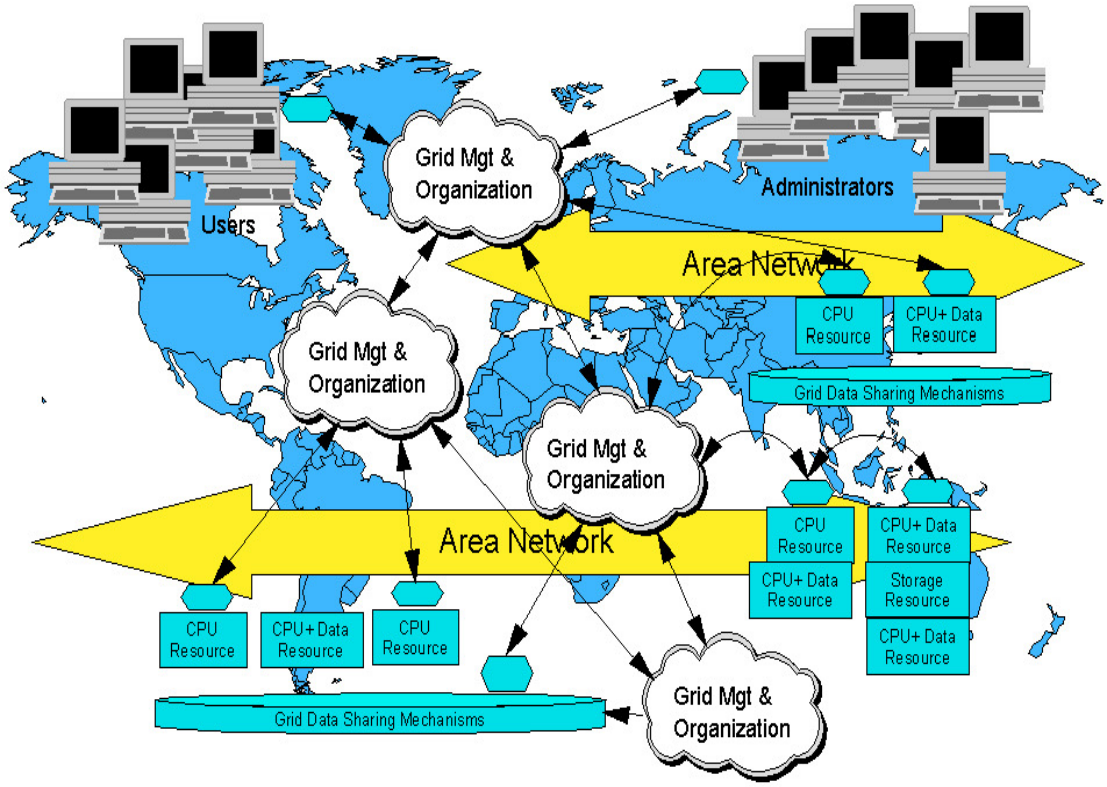

Figure 1.3: View of Inter-grid spanning over the globe. [5]

Detailed discussion about Inter-grid can be found in [7], computational grid is very similar to electric grid. When we require additional electricity we have to just plug into a power grid to access additional electricity on demand, similarly we plug into a computer grid to access additional computing power on demand using the cheapest possible resource. Basically, user submits a job to the grid through an interface on his computer that serves as a portal to the grid. Special grid management software accepts the job and breaks it down into hundreds or even thousands of independent tasks, then locates idle processors and distributes the tasks among them. Finally, it aggregates the works and spits out the results.

1.2 Resource Management Systems for Computational Grids

The theory behind the grid computing is not to buy more computational resources but to borrow the computing power from where it is not being used. The more are the resources the more complex is their management. Resource management in the computational grids deal with identifying requirements, matching resources to the applications, allocation of resources and scheduling and monitoring of the grid in a timely manner. Security and fault tolerance along with scheduling makes the process of resource management challenging for grid systems. The other factors like site autonomy and resource heterogeneity add more complexity to this system [8]-[11].

In the traditional resource management systems, the assumption is that they have complete control on the resource. Thus effective mechanisms and policies can be implemented for efficient use of that resource. But in Grid systems resources are distributed across separate administrative domains. This results in resource heterogeneity, differences in usage, scheduling policies, security mechanisms. Co-allocation of the resources also becomes complex due to site autonomy. Co-allocation is the problem of allocating resources in different sites to an application simultaneously. The Figure 1.4 shows the high level view of a Grid Resource management system. In a grid system, an end user submits to the management system the job to be executed along with some constraints like job execution deadline, the maximum cost of execution. The function of the resource management is to take the job specification and from it estimate the resource requirements like the number of processors required, the execution time, and memory required. After estimating the resource requirements RMS is responsible for discovering available resources [11] and selecting appropriate resources for job execution. Finally schedule the jobs on these resources by interacting with the local resource management system.

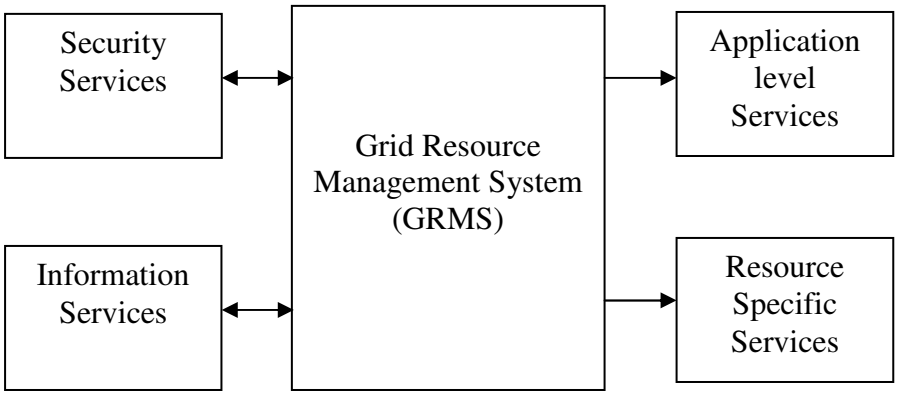

Figure 1.4: High level view of Grid Resource Management System.

Hence the major functions of the resource management systems include accepting the user jobs along with the specified constraints discover a set of matching resources that can execute the job respecting all the constraints and finally schedule the jobs on to the resource.

1.2.1 Grid Resource Discovery

As described earlier, grid environment consists of large number of shared resources distributed among the many locations. Discovery of suitable resources become one of the major components of the Resource Management i.e. performed on the distributed resources, applications etc. Resource discovery is an information-gathering process that digs deep into the details of application requirements from the perspective of client's business, target audience, and industry. Resource Discovery is a very important process because it allows you to see when and where things happened and to collect the information necessary to assess against your desire. Resource Discovery can be defined as a directory service directed to the spontaneous networking environments [12]. Discovery in the grid environment becomes complex as the resources are geographically distributed, heterogeneous in nature and owned by

different individuals and organizations each having their own resource management policies and different access and cost models. The problem of discovering grid resources can be defined as the problem of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements [13]. Resource discovery is performed by resource management system to obtain information about the available resources. In these networks, resources can enter or leave at any time and this mechanism aims to provide information about the resources available at a specific moment [14]. Resource Discovery deals with finding, locating, searching and organizing the matching data (information, facts, quires, numbers etc.), in response to queries of users or an automated mechanism with following trades-off:

- ✓ In minimal possible time
- ✓ Respecting resources' access policy constraints and standards.
- ✓ At minimal cost to the user

Resource Discovery systems become more and more important for highly distributed systems like grid, as resources may leave or join at any time. According to Koen et al. Resource Discovery systems can be categorized by different design aspects mentioned in the [15]. Each of these aspects represents a design choice for which several solutions are possible. The design aspects are: service provider, construction, foreknowledge, architecture, registration, discovery, supported resources, naming and queries.

Resource Discovery has generally two key steps [16]. The first step is to locate a resource, and the second step is to connect or match to it. Locating resource means getting a query to metadata or data. Resource locating would not be quite difficult in a local network that is managed by a central server. Traditional lookup service, i.e.

LDAP service, maintains all information, such as name, location, attribute in a single server that is normally based on central DBMS, which will result constant search performance. But it becomes variable when arbitrary joins and leaves are allowed. The second step after locating the resources is matching that means determining a match between user-specified query and resource characteristics.

1.2.2 Grid Resource Discovery Models

Grid resource discovery mechanisms need to update the resource information regularly. Based on how frequently the information is refreshed the resource discovery models can be categorized into Periodic and On-demand updating models.

1.2.2.1 Periodic Dissemination Model

Periodic models update the information about the recourses' availability after a regular interval of time. They can be further categorized as pull or push models based on who initiates the information update process. In the pull model, resource information system collects data about the resources regularly and updates its database. On the other hand in the push model, resources itself update its information in the database after a regular interval of time.

1.2.2.2 On – Demand Dissemination Model

On-Demand dissemination models are different from periodic models in a way that they don't update the information regularly. Instead the information is updated when an event occurs or some status change occurs.

1.2.3 Grid Resource Scheduling

Another challenging task in the grid resource management is the process of scheduling applications over Grid resources [17]. A Grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and

usually owns the resource. A Grid scheduler is in charge of resource discovery, Grid scheduling (resource allocation and task scheduling), and job execution management over multiple administrative domains.

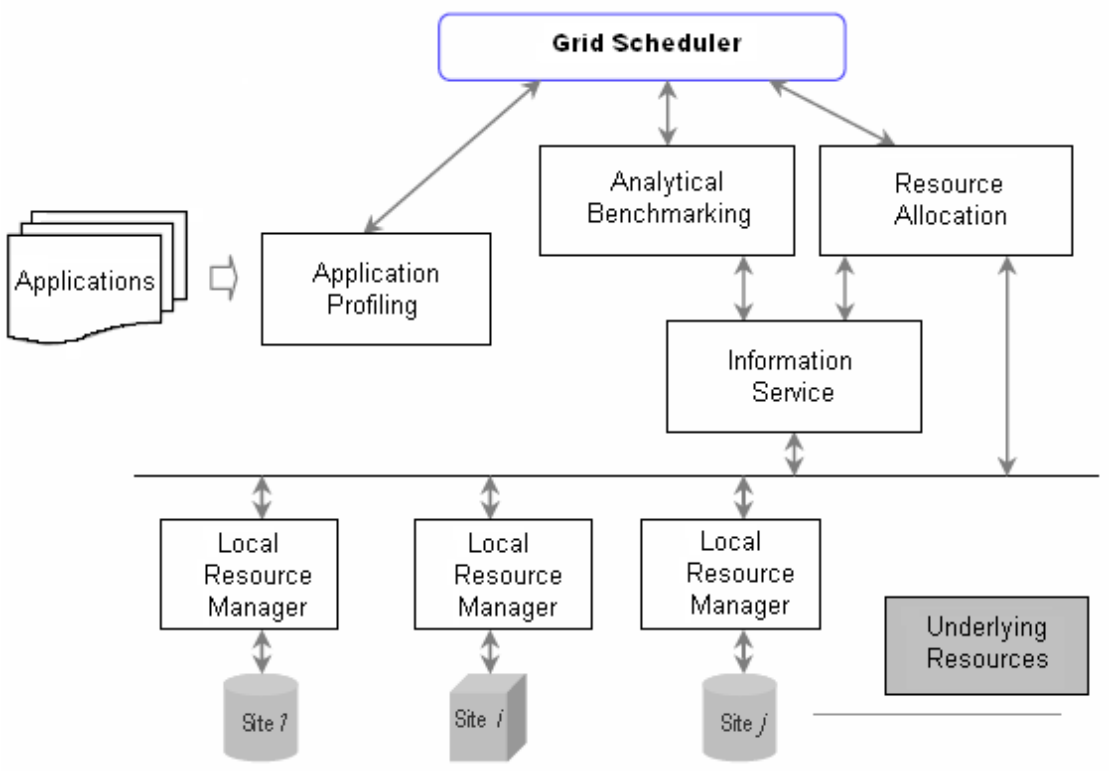

Figure 1.5: Common Grid Scheduler Architecture

Common grid scheduler architecture is shown in Figure 1.5. Local resources lie at the bottom of the architecture each being managed by a local resource manager (LRM). LRM enables low level processing of requests and executing a job that satisfy a request, remote monitoring and management of the jobs as well as updating the information service with the current availability and capabilities of the resources that it manages. Information service component maintains the various resource characteristics and answers to the queries for acquiring the information about the resources. An adequate scheduler must incorporate the current information of resources when making scheduling decisions. Application profiling interface provides a way for user to submit jobs to the grid where as the analytical benchmarking

component provides a measure of how well a computational resource will perform on a given job. Grid Scheduler (sometimes also known as broker) is most important of this architecture. It has two prime responsibilities: one is resource selection, which is the process of selecting feasible and available resources for a given application. Other prime task is mapping which is the process of placing the jobs on to resources for execution. Resource Allocation (RA) component implements a finally-determined schedule through allocating the resources to the corresponding jobs.

With a common Grid infrastructure, Grid scheduling is to answer the question: Given a set of Grid applications (usually computing intensive), how to schedule them over multiple decentralized resources? Each application has a number of tasks. In mapping tasks to resources, we have several basic questions to deal with:

- ✓ How do the relations between tasks affect scheduling decisions?
- ✓ How does the heterogeneity of resources affect the performance of a schedule?
- ✓ What performance models should a scheduler use to determine the quality of a schedule?

1.2.4 Grid Resource Scheduling Models

Grid scheduling is intrinsically more complicated than local resource scheduling because it must manipulate large-scale resources across management boundaries. In such a dynamic distributed computing environment, resource availability varies dynamically [17]. So scheduling becomes quite challenging. Research in the resource scheduling focuses mainly on these four areas.

1.2.4.1 Static Task scheduling

Given a set of tasks and a set of resources, a static scheduler computes the execution schedule before run time. In static task scheduling, resource information and performance parameters are assumed to be known. Depending on how a job can

be divided, relevant research can be categorized into two different areas: divisible workload scheduling [18]-[20], where they can divide workload into arbitrary-sized pieces ("chunks"), and fixed-sized independent tasks scheduling [21].

1.2.4.2 Application-level Scheduling

Application-level scheduling, which explores the effect of different application requirements on scheduling and suggests adaptive scheduling mechanisms based on runtime resource availability. The idea of application-level scheduling is to embed scheduling behavior into the application itself to make application aware of resource variations so that it could decide work size for each job piece based on dynamic resource availability. Early research in this area is quite application-coupled [22] [23].

1.2.4.3 Resource Availability Prediction

The most commonly requested resources by a Grid application requests are computing, data, and network resources. For a Grid resource, it is easy to get a machine's static resource information, such as CPU frequency, memory size, network bandwidth, file systems, etc. But application run-time resources, such as CPU load, available memory, and available network capacity, are variable because Grid is a dynamic resource-sharing computing environment in which there are many applications running. At time of scheduling decision making, accurate resource predication on run-time resource measurement parameters is critical to maximize application performance. In predication research, the analysis is based on historical data on previous resource availability information and job performance records [24]. Various statistical techniques can apply. If resource behavior follows or is assumed to follow certain distribution, stochastic process analysis can be used to predict future resource measurements on a fixed time point or during a certain interval of time.

Regression technique can be used for performance prediction in presence of a performance model.

1.2.4.4 Economic methods in Decentralized Task Scheduling System

The nature of Grid as a large-scale distributed system complicates the scheduling strategies and algorithms. But the cost of scheduling itself must be reasonable to make it efficient. As we look at Grid as a decentralized system, we may think of making the scheduling process decentralized as well. In research on decentralized scheduling, economic theories in market are taken into consideration, because, in nature, market is also a decentralized dynamic system dealing with competitive resources. In market, price is the only measurement while operating. If a pricing policy can be decided for producer (resource) and consumer (applications), using price reduces much communication cost during the process of scheduling. Current market-oriented programming models are based on general equilibrium theory. Given some assumptions, the general equilibrium theory can be transformed to optimization problems, which a scheduling problem is to solve [25]-[28].

Scheduling multiprocessor tasks with unit processing times is a strongly NP-hard problem, which makes it a candidate for optimization.

1.3 Meta-Heuristic Techniques

A meta-heuristic is a heuristic method for solving a very general class of computational problems by combining user-given black-box procedures (usually heuristics themselves) in the hope of obtaining a more efficient or more robust procedure [29]. The name combines the Greek prefix "meta" ("higher level") and "heuristic" ("to find").

Local Search is an iterative transition process that starts with an initial solution and produces a sequence of solutions by applying small local changes. If a new solution rates better when compared with the best current solution, it replaces the current best solution. The whole searching process halts upon reaching the maximum number of iterations, or when no improvement seems possible. The solution to the problem is the best solution found so far. Local search works well when the problem is linear, because in those cases, the local optimum is also the global optimum. However, in non-linear cases where there are several local optimums, a bad starting point can cause local search to terminate upon encountering the first local optimum. This problem is illustrated in diagram. In Figure 1.6 the global optimum *y* is the solution. If the local search algorithm starts at point *a*, then the solution *x* is returned because it is the local optimum. However, if the algorithm were to start from *b* then we are guaranteed to get *y* returned as the solution. Thus, a good starting point plays an important role in local search, but to overcome local optimality one has to switch to meta-heuristics.

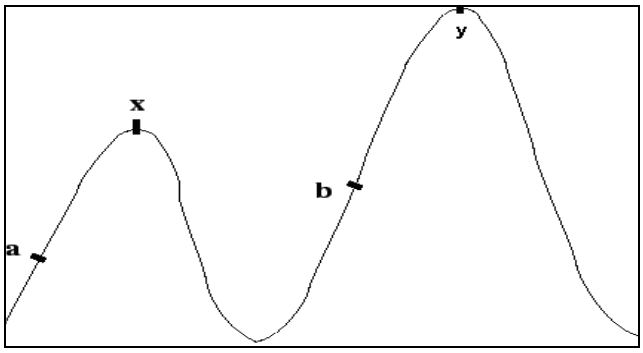

Figure 1.6: Local Optimum Problem

Meta-heuristic is a collection of heuristic algorithms applicable to a wide variety of problems. The heuristic approach is often associated with ``rules of thumb'' or clever insights. Based on the heuristic provided, the algorithm performs the search process iteratively to look for a solution. The iterative search process is terminated

when no improvements are possible. The choice of meta-heuristic algorithm depends on a few factors, such as the solution quality required and the availability of problem knowledge. Because meta-heuristic is a rather ad hoc technique, there is no guarantee a solution can be obtained. In most cases, meta-heuristic techniques served as the last resort when the other two techniques could not produce the desired solutions. Some of the meta-heuristics like Genetic Algorithms (GA), Simulated Annealing (SA) and Ants' Colony Optimization (ACO), used to tackle the problem of resource management in Computational grids are discussed below.

1.3.1 Simulated Annealing (SA)

Simulated annealing is a probabilistic search algorithm. The term simulated annealing derives from the process of heating and then cooling a substance slowly to finally arrive at the solid state [30]. The search algorithm simply mimics the physical process as follows.

In the early stages of the execution, the temperature is high resulting in higher probability, and cause jumping to occur more frequently. Jumping occurs as a way of avoiding local minima, accepting a poorly performed solution with a higher probability. Otherwise, any solution performing better than the current solution is accepted and replaced as the best solution found so far. As the execution time elapses, the temperature decreases thus reduces the frequency of jumping. This probabilistic nature of the system guarantees the exploration of other solution space instead of terminating whenever encountering the first local optimum.

The simulation process terminates after a number of successive executions with no improvements, and returns the best solution so far. The only drawback of simulated annealing is the long execution time to obtain quality solutions. Although it is possible to achieve global optimum solution, it comes at a cost of slower cooling

procedure and longer iteration at each temperature level. Conversely, if in favour of shorter execution time, the algorithm compromises the solution quality.

1.3.2 Genetic Algorithms (GA)

GAs are adaptive methods that can be used to solve optimization problems [31], based on the genetic process of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and *Survival of the Fittest*, first clearly stated by *Charles Darwin* in *The Origin of Species*. By mimicking this process, GAs are able to "*evolve*" solutions to real world problems, if they have been suitably encoded. GA search is constrained neither by the continuity of the function under investigation, nor the existence of a derivative function. It is assumed that a potential solution to a problem may be represented as a set of parameters. These parameters (known as genes) are joined together to form a string of values (known as a chromosome). A fitness function must be devised for each problem to be solved. Given a particular chromosome, the fitness function returns a single numerical fitness or figure of merit, which will determine the ability of the individual, which that chromosome represents. Reproduction is another critical attribute of GAs where two individuals selected from the population are allowed to mate to produce offspring, which will comprise the next generation. Having selected two parents, their chromosomes are recombined, typically using the mechanisms of crossover and mutation. Traditional view is that crossover is the more important of the two techniques for rapidly exploring a search space. Mutation provides a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined. If the GA has been correctly implemented, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases towards the global optimum.

Parallel and distributed algorithms based on simulated annealing and genetic algorithms are applied for the optimization of test vector generation in VLSI circuits [32][33], which is also an NP- complete problem.

1.3.3 Ant Colony Optimization (ACO)

Compared to humans, an individual ant has very little brainpower. While humans have approximately 10 billion neurons, ants have only 250,000. The real power of ants resides in their colony brain. The self-organization of those individuals is very similar to the organization found in brain-like structures. Like neurons, ants use mainly chemical agents to communicate; one ant releases a molecule of pheromone that will influence the behavior of other ants [34]. Ant algorithms are often compared with other evolutionary approaches such as Genetic Algorithms, Evolutionary Programming and Simulated Annealing. It is important to remember that Ant algorithms are non-deterministic and rely on heuristics to approximate to a sub-optimal solution in cases where the number of combinations is extremely huge and impossible to calculate using a deterministic algorithm.

Simulation of ant's behaviour in computer system is a kind of parallel search over several constructive computational threads. Concurrent and asynchronous set of computational agents, known as colony of ants, move through the states of the problem corresponding to its partial solution. The decision to move to next state is influenced by pheromone trail on the route and heuristic information. Each ant constructs the solution during its tour. Pheromone trails are updated at the completion of tour, which will direct the search of future ants.

1.4 Thesis Objectives and Contributions

As observed in the previous sections, grid resource management is a compute intensive problem. Two major areas that need focus are Grid Resource discovery and Grid Scheduling. Given a set of independent tasks and a set of available resources, an efficient grid resource discovery mechanism needs to find the matching resources in a minimal span of time and a grid scheduler attempts to minimize the total execution time of the task set by finding an optimal mapping of tasks to machines. The metric used to find such a mapping is the estimate of "turnaround time" or completion time (machine available time + expected time to compute).

Finding the best mapping is actually a combinatorial optimization problem. Grid resource management is further made challenging by dynamic availability of the resources. The common solution is to use some heuristic search procedure to find a near-optimal solution quickly. Most of the existing research in this area is quite theoretical, because even heuristic search is often too expensive for large search spaces. In a Grid, however, due to the existence of large number of tasks and resources, a practical and efficient meta-heuristic based search algorithm can be a desirable solution for finding the set of matching resources to a user query and then mapping the resource to applications/tasks. Also, more practical experiments should be done to verify the results under a broad range of machine conditions.

In this thesis, therefore, various Grid Resource Management Systems and Meta-heuristic techniques (like GA, SA and ACO) are explored and different ways are described to optimize the resource management in computational grids by using meta-heuristic search algorithms.

1.4.1 Objectives of the Thesis

The objectives of the thesis are:

- ✓ To study and explore Grid Resource Management Systems.
- ✓ To study and analyze Meta-Heuristic techniques (Simulated Annealing & Genetic Algorithms or some hybrid techniques based on these) that can be used for resource optimization.
- ✓ To explore the possibility of using software Agents in Grid Resource Management system.
- ✓ To demonstrate the usability of proposed techniques in Grid Resource Management, so as to provide better Quality of Service. (QoS)

1.4.2 Contributions

To support the thesis "Meta-heuristic techniques for Grid Resource Management" and to demonstrate that meta-heuristics based search algorithms can deliver significant value in order to optimize the grid resource management; several novel research contributions have been made in this work. These are as follows:

- ✓ A detailed study of some of existing resource management systems has been done and Quality of Service (QoS) parameters are identified.
- ✓ An objective function has been formulated, taking the QoS parameters (as identified in first objective) into considerations, which tries to take care of user specified deadlines, as well as tries to ensure the maximum utilization of resources.
- ✓ Grid scheduling has been formulated as a search problem and meta-heuristics like GA, SA, ACO or some hybrid of these techniques has been tried to find the optimal solution. Three new meta-heuristic based grid-scheduling algorithms, as given below, have been proposed, implemented and validated:

- SGASchedule: Proposed SexualGA based Resource Scheduling Algorithm
- ACOSchedule: Proposed ACO based Resource Scheduling Algorithm
- HybridGSA: Proposed Genetic Simulated Annealing based Hybrid Resource Scheduling Algorithm

✓ A software mobile agent based decentralized approach for grid resource discovery has been proposed. Also, a new ACO based algorithm has been proposed that tries to find a set of resources matching to the user query in a quick and efficient manner. Software mobile agents are the components that actually travel to search for resources and ACO based algorithm guides the mobile agent's travel to find the matching resources.

✓ To demonstrate the usability of proposed techniques, a Grid Simulator has been designed and implemented based on well-known grid simulation toolkit known as GridSim. This Grid Simulator implements a centralized scheduler and provides an easy way to integrate and test new scheduling algorithm.

✓ To demonstrate how the above-proposed techniques behave in actual networked environment, a Grid Test bed has been designed and implemented. This test bed provides a grid like environment with web-based user interface. This environment can be used for testing and validating algorithms for grid resource management.

✓ The experimental results prove that proposed algorithms perform better as compared to the existing algorithms. The existing algorithms for resource management mainly focus on the optimizing the resource utilization where as the proposed algorithms try to optimize the resource utilization as well as the

cumulative time delay. Hence the proposed algorithms take care of the interests of both: the resource providers and the grid users.

1.5 Thesis Organization

The thesis is organized as follows.

Chapter 1: Introduction: This chapter provides the basic introduction to the Grid Computing, types of grid and benefits of grid. It provides an overview of Grid Resource Management System. Basic introduction to the meta-heuristics used for optimization and Mobile Agents is also given in this chapter.

Chapter 2: Literature Review: This chapter provides the literature review of earlier work done. A comparative study and analysis of the existing grid resource management systems has also been provided in this chapter.

Chapter 3: Proposed Algorithms for Grid Resource Scheduling: This chapter discusses in detail, three new algorithms for grid resource scheduling proposed in this thesis. This chapter provides the objective function formulation that takes care of the interests of both: resource providers and resource users. Three algorithms are proposed. These are

1. Proposed SexualGA based Resource Scheduling Algorithm (SGASchedule),
2. Proposed ACO based Resource Scheduling Algorithm (ACOSchedule),
3. Proposed Hybrid algorithm for Resource Scheduling based on Genetic Simulated Annealing (HybridGSA).

Details about these proposed algorithms are discussed in this chapter.

Chapter 4: Proposed algorithm for Grid Resource: This chapter provides the details of proposed resource discovery algorithm for grids. This algorithm uses mobile agents for resource discovery. To guide the mobile agents in resource discovery, An ACO based algorithm (ACORD) has been proposed, implemented and tested. This

ACO based algorithm provides learning to the mobile agents, while they travel from one node to another for resource discovery.

Chapter 5: Design and Implementation of Proposed Algorithms: This chapter provides the design and implementation details of the all four proposed algorithms (three for grid resource scheduling and one for grid resource discovery). Grid scheduling algorithms are implemented on the grid simulator based on the GridSim. GridSim allows modeling and simulation of entities in parallel and distributed computing systems for design and evaluation of scheduling algorithms.

Also, the implementation of scheduling and resource discovery algorithms is done on Grid Test bed. Grid Test bed provides the actual grid like environment with a simple web based user interface for submission of jobs, registering and de-registering of resources and administrative tasks. This testbed provides inconsistent and partially consistent completion time (CT) matrix based prediction system which is based on the simulation model used in [21] for the comparison of scheduling heuristics.

Chapter 6: Experimental Details and Results: This chapter provides the details of different experimental scenarios and results of algorithms and techniques proposed in this thesis. Experiments are done on GridSim based scheduler to evaluate proposed algorithms on large-scale grids in a controlled manner. Comparisons have been made with Nimrod/G implementation on GridSim. Experiments are performed on Grid Test Bed to compare the proposed algorithms existing scheduling algorithms based on GA, ACO and SA. This chapter also provides the experimental details and results of the ACO guided mobile agents based resource discovery algorithm.

Chapter 7: Conclusion and Future Scope: Finally, this chapter provides the conclusions and future scope of the work. It is clear from the experimental results that proposed meta-heuristic based algorithms are better to tackle the problem of resource

management in grid environment as compared to the existing ones. The future research can be focused on the parallel implementation of the proposed algorithms.

1.6 Summary

This chapter starts with an introduction and motivation of grid computing followed by different types of grid and its benefits. A brief discussion on the Grid resource management system is provided. The major problems of grid resource management like grid scheduling and grid resource discovery are introduced. An overview of different meta-heuristics is also provided. Finally this chapter provides the problem formulation and objectives of the thesis followed by the thesis contributions and thesis organization. Next chapter focuses on the study of existing Grid Resource Management systems and existing algorithms for grid scheduling and grid resource discovery.