CHAPTER 1

INTRODUCTION

1.1 GRID COMPUTING

Nowadays, the scientific problem becomes very complex; therefore, it requires more computing power and storage space. These requirements are very common in an organization while dealing with current technological data and requirements. Based on these basic requirements, they also need higher computational resources when dealing with current technological methodology. The past technologies such as distributed computing and parallel computing are not highly suitable for the recent advancements because, the modern computer industry is operating very large amounts of data which is geographically, distributed. Abrishami et al. (2012) analysed the processing speed, computational power and other resources in distributed and parallel computing environments. So, in the past few years, the grid computing has been proposed as an effective resource management in order to meet the organizational requirements. Figure 1.1 shows a historic development of grid computing from networking and processing aspects.

Grid computing is a growing technology, and is a High Performance Computing (HPC) branch for solving complex problems. Banerjee et al. (2010) proposed various computing components such as a traditional supercomputing centre, clusters of computers, a heterogeneous network, a distributed resources centre which are integrated in a hardware and software infrastructure.
Grid computing has an architecture similar to distributed computing but it is distinguished from almost all distributed computing paradigms by the following characteristics: the essence of grid computing lies in the efficient and optimal utilization of a wide range of heterogeneous, loosely coupled resources in an organization tied to a sophisticated workload management capabilities or information virtualization.

Grid computing is a virtualized distributed computing environment which aims at enabling the dynamic “runtime” selection, sharing, and aggregation of distributed autonomous resources based on the availability, capability and performance. The generalized architecture of grid shown in Figure 1.2, was proposed by Boukkerram and Azzou (2006), and modified by Somasundaram and Radhakrishnan (2008), which is a grid broker that handles the file transfer optimization, fault tolerance by multithreading and push and pull task assignment. The virtual organization defined by Foster et al. (2001) using grid computing is shown in Figure 1.3.
Figure 1.2 Architectural design of grid computing

Figure 1.3 Virtual organization based grid architectural design
1.2 COMPONENTS AND CONCEPTS OF GRID COMPUTING

In this section, various components of grid computing are introduced, as shown in Figure 1.4.

![Figure 1.4 Components of grid computing](image)

1.2.1 Types of Resources

A grid is a collection of machines, sometimes referred to as “nodes,” “resources,” “members,” “donors,” “clients,” “hosts,” “engines,” and many other such terms. They all contribute any combination of resources to the grid as a whole. Some resources may be used by all users of the grid while others may have specific restrictions. In Figure 1.5, the organizations having resources are interconnected thorough grids computing. The data flow in the grid computing and the functionalities with real time example are shown in Figure 1.6.
Figure 1.5 Resources of world wide organization with grid computing

Figure 1.6 Data flow in grid computing with real time example
1.2.2 Computation

The first common resource in grid is computing cycles, provided by the processors of the machines on the grid, which was proposed by Xiao et al. (2008) and Sonmez et al. (2010). The processors can vary in speed, architecture, software platform, and other associated factors such as memory, storage, and connectivity. There are three primary ways to exploit the computation resources of a grid. The first and simplest is to use it to run an existing application on an available machine on the grid rather than locally. The second is to use an application designed to split its work in such a way that the separate parts can execute in parallel on different processors. The third is to run an application, which needs to be executed many times, on many different machines in the grid.

Scalability is a measure of efficient use of multiple processors on a grid. If twice as many processors make an application complete in one half the time, then it is said to be perfectly scalable. However, there may be limits to scalability when applications can only be split into a limited number of separately running parts or if those parts experience some other contention for resources of some kind.

1.2.3 Storage

The second most common resource used in a grid is data storage. A grid providing an integrated view of data storage is sometimes known as data grid. Each machine on the grid usually provides some quantity of storage for grid use, even if temporary. Storage can be memory attached to the processor or it can be secondary storage using hard disk drives or other permanent storage media. Memory attached to a processor is usually very fast but is volatile. It would best be used to cache data or to serve as temporary storage for running applications.
Secondary storage in a grid can be used to increase capacity, performance, sharing, and reliability of data. Many grid systems use mountable networked file systems, such as Andrew File System, Network File System, Distributed File System, or General Parallel File System. Various file systems and job scheduling on these file systems are available in Berten et al. (2006), and Lee and Zomaya (2007) which offers varying degrees of performance, security features, and reliability features. Capacity can be increased by using storage on multiple machines with a unifying file system. Any individual file or database 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, resulting in an easier use of reference data residing in the grid, without regard for its exact location.

Similarly, special database software can “federate” an assortment of individual databases and files to form a larger, more comprehensive database, accessible using database query functions. More advanced file systems on a grid can automatically duplicate sets of data, to provide redundancy for increased reliability and performance. An intelligent grid scheduler can help to select an appropriate storage device to hold data, based on usage patterns. Then, jobs can be scheduled closer to the data, preferably on the machines directly connected to the storage devices holding the required data.

When there are sequential or predictable access patterns to data, data striping technique can create the virtual effect of having storage devices that can transfer data at a faster rate than any individual disk drive. This is important for multimedia data streams or when collecting large quantities of data at extremely high rates from scans or particle physics experiments, for example. A grid file system can also implement journaling so that data can be recovered more reliably after failures. In addition, some file systems implement advanced synchronization mechanisms to reduce contention when
data is shared and updated by many users. Figure 1.7 shows the storage of grid computing with sharing mechanism.

Figure 1.7 Grid computing with storage and sharing mechanism

1.2.4 Communications

Figure 1.8 Communication in grid computing
The rapid growth in communication capacity among machines today makes grid computing practical, compared to the limited bandwidth available when high speed distributed computing first emerged. Therefore, it should not be a surprise that another important resource of a grid is data communication capacity. This includes communications within the grid and outside the grid. Figure 1.8 represents the communication in the real inter grid architecture.

Communications within the grid are important for sending jobs and their required data to nodes within the grid, as was proposed by Wang et al. (2009) and Gaeta et al. (2011). Some jobs require a large amount of data to be processed and it may not always reside on the machine running the job. The bandwidth available for such communications can often be a critical resource that can limit utilization of the grid.

Access to the Internet, for example, can be valuable when building search engines. Machines on the grid may have connections to the external Internet in addition to the connectivity among the grid machines. When these connections do not share the same communication path, then they add to the total available bandwidth for accessing the Internet.

Redundant communication paths are sometimes needed to better handle potential network failures and excessive data traffic. In some cases, higher speed networks must be provided to meet the demands of jobs transferring a larger amount of data. A grid management system can better show the topology of the grid and highlight the communication bottlenecks. This information can in turn be used to plan for hardware upgrades.
1.2.5 Software and Licenses

The grid may have software installed that may be too expensive to install on every grid machine. Using a grid, the jobs requiring this software are sent to the particular machines on which this software is installed. When the licensing fees are significant, this approach can save significant expenses for an organization.

Some software licensing arrangements permit the software to be installed on all of the machines of a grid but may be limited to the number of installations that can be simultaneously used at any given instant. License management software keeps track of how many concurrent copies of the software are being used and prevents more than that number from executing at any given time. The grid job schedulers can be configured to take software licenses into account, optionally balancing them against other priorities or policies.

1.2.6 Special Equipment, Capacities, Architectures, Policies

Platforms on the grid will often have different architectures, operating systems, devices, capacities, and equipment. Each of these items represents a different kind of resource that a grid can use as criteria for assigning jobs to machines. While some software may be available on several architectures, for example PowerPC and x86, is often designed to run only on a particular type of hardware and operating system. Such attributes must be considered when assigning jobs to resources in the grid.

In some cases, the administrator of a grid may create a new artificial resource type that is used by schedulers to assign work according to policy rules or other constraints. For example, some machines may be designated to be used only for medical research. These would be identified as
having a medical research attribute and the scheduler could be configured to assign only jobs that require machines of the medical research “resource”. In such situation, jobs requiring a “military resource” would not be assigned to such machines. Of course, the administrators would need to impose a classification on each kind of job through some certification procedure to use this kind of approach.

1.2.7 Enrolling and Installing Grid Software

Enrolling in grid may require authentication for security purposes. The user positively establishes his identity with a Certificate Authority. This should not be done solely via the Internet. The Certificate Authority must take steps to assure that the user is, in fact, the user claims to be. The Certificate Authority makes a special certificate available to software, needs to check the true identity of a grid user and his grid requests. Similar steps may be required to identify the donating machine. The user has the responsibility of keeping his grid credentials secure.

Once the user and/or machine are authenticated, the grid software is provided to the user for installing on his machine for the purposes of using the grid as well as donating to the grid. This software may be automatically preconfigured by the grid management system to know the communication address of the management nodes in the grid and user or machine identification information. In this way, the installation may be a one click operation with a minimum of interaction required on the part of the user. In less automated grid installations, the user may be asked to identify the grid’s management node and possibly other configuration information. The user may choose to limit the resources donated to the grid, the times that his machine is usable by the grid, and other policy related constraints. The user may also need to inform the grid administrator.
1.2.8 Logging onto the Grid

To use the grid, most grid systems require the user to log on to a system using a user ID that is enrolled in the grid. Other grid systems may have their own grid login ID separate from the one on the operating system. A grid login is usually more convenient for grid users. It eliminates the ID matching problems among different machines. To a user, it makes the grid look more like one large virtual computer rather than a collection of individual machines. Globus, for example, implements a proxy login model that keeps the user logged in for a specified amount of time, even if the user logs off, and even if the machine is rebooted.

Once logged on, the user can query the grid and submit jobs. Some grid implementations permit some query functions if the user is not logged into the grid or even if the user is not enrolled in the grid.

1.2.9 Queries and Submitting Jobs

The user will usually perform some queries to check to see how busy the grid is, to see how the submitted jobs are progressing, and to look for resources on the grid. Grid systems usually provide command line tools as well as graphical user interfaces (GUIs) for queries. Command line tools are especially useful when the user wants to write a script that automates a sequence of actions. For example, the user might write a script to look for an available resource, submit a job to it, watch the progress of the job, and present the results when the job has finished.

Job submission in grid usually consists of three parts, even if there is only one command required, which is presented in Korkhov et al. (2009). First, some input data and possibly the executable program or execution script file are sent to the machine to execute the job and is called “staging the input
data.” Alternatively, the data and program files may be pre-installed on the grid machines or accessible via a mountable networked file system. When the grid consists of heterogeneous machines, there may be multiple executable program files, each compiled for the different machine platforms on the grid.

A useful feature provided by some grid systems is to register these multiple versions of the program so that the grid system can automatically choose a correct matching version to the grid machine that will run the program. Some grid technologies require that the program and input data be first processed or “wrappered” in some way by the grid system. This may be done to add protective execution controls around the application or just to collect all of the data files into one.

Second, the job is executed on the grid machine. The grid software running on the donating machine executes the program in a process on user’s behalf. It may use a common user ID on the machine or it may use the user’s own user ID, depending on which grid technology is used. Some grid systems implement a protective “sandbox” around the program so that it cannot cause any disruption to the donating machine if it encounters a problem during execution. Rights to access files and other resources on the grid machine may be restricted.

Third, the results of the job are sent back to the user who submits the job. In some implementations, intermediate results can be viewed by the user who submitted the job. In some grid technologies that do not automatically stage the output data back to the user, the results must be explicitly sent to the user, perhaps using a networked file system.

Scripts are also useful for submitting a series of jobs, for a parameter space application. Some computation problems consist of a search for the desired result based on some input parameters. The goal is to find the
input parameters that produce a best desired result. Bossenbroek et al. (2009) proposed a scheduling, in which, for each input parameter a separate job is executed to find the result for that value. The whole application consists of many such jobs, which explore the results for a large number of input parameter values.

Scripts are usually used to launch the many sub-jobs, each receiving its own specific parameter values. Parameter inputs can sometimes be more complex than simply a number. Sometimes a different input data set represents different “input parameter.” Scripts help to automate a large variety of more complex parameter space study problems. For simpler parameter space inputs, some grid products provide a GUI to submit the series of sub-jobs, each with different input parameter values, as presented in Rajah et al. (2009).

When there are a large number of sub-jobs, the work required to collect the results and produce the final result is usually accomplished by a single program, usually running on the machine at the point of job submission. If large number sub-jobs are required for an application, the work of collecting the results might be distributed as well. For example, the sub-job that submits more sub-jobs to the grid would be responsible for collecting and aggregating the results of the sub-jobs it spawned.

1.2.10 Data Configuration

The data accessed by the grid jobs may simply be staged in and out by the grid system. However, depending on its size and the number of jobs, this can potentially add up to a large amount of data traffic, as brought out by Lee et al. (2009). For this reason, attention is usually given on how to arrange to have the minimum of such data movement on the grid. For example, if there is very large number of sub-jobs running on most of the grid systems for
an application that will be repeatedly run, the data they use may be copied by each machine and reside until the next time the application runs. It is preferable to use a networked file system to share this data, because in such a file system, the data would be effectively moved from a central location every time the application is run. This is true unless the file system implements a caching feature or replicates the data automatically.

There are many considerations in efficiently planning the distribution and sharing of data on a grid. This type of analysis is necessary for large jobs to utilize the grid better and avoid creating unnecessary bottlenecks.

1.2.11 Monitoring Progress and Recovery

The user can query the grid system to see how the application and its sub-jobs progress. The sub-jobs are proposed by many authors and proved reliable, as seen in the attempts by Amoroso and Marzullo (2006), and Levitin et al. (2006). When the number of sub-jobs becomes large, it is too difficult to list them all in a graphical window. Instead, there may simply be a one large bar graph showing some averaged progress metric. It becomes very difficult for the user to tell if a particular sub job is not running properly. A grid system, in conjunction with its job scheduler, often provides some degree of recovery for sub jobs that fail. A job may fail, as identified by Dai et al. (2007), due to the following reasons:

- Programming error: The job stops in the middle of execution with some program fault.
- Hardware or power failure: The machine or devices being used stop working in some way.
- Communications interruption: A communication path to the machine has failed or is overloaded with other data traffic.
• Excessive slowness: The job might be in an infinite loop or normal job progress may be limited by another process running at a higher priority or some other form of contention.

It is not always possible to automatically determine the reason for a job’s failure due to a problem with the design or it is due to failures of various kinds in the grid system infrastructure. Schedulers are often designed to categorize job failures in some way and automatically resubmit jobs, as elaborated by Bucur and Epema (2007), so that they are likely to succeed, running elsewhere on the grid. In some systems, the user is informed about any job failures to decide whether to issue a command to attempt to rerun the failed jobs.

Grid applications can be designed to automate the monitoring and recovery of their own sub-jobs using functions provided by the grid system software Application Programming Interfaces.

1.2.12 Reserving Resources

To improve the quality of a service, the user may arrange to reserve a set of resources in advance for exclusive or high priority use. A calendaring system analogy can be used here. Such a reservation system can also be used in conjunction with planned hardware or software maintenance events, when the affected resource might not be available for grid use.

In some grid system, it may not be possible to reserve specific machines in advance. Instead, the grid management systems may allocate a larger fraction of its capacity for a given reservation to allow for the likelihood of some of the resources becoming unavailable. This must be done in conjunction with tools that have profiled the grid’s workload capacity sufficiently to have reliable statistics about the grid’s ability to serve the reservation.
1.3   RECENT DEVELOPMENTS IN GRID COMPUTING

In this section, various recent implementation of grid based computing and its major tasks are briefly described.

The Australian Partnership for Advanced Computing (APAC) Grid interconnects various grid sites distributed across Australian Institutions and Universities. The APAC Grid uses a hierarchical information service, MDS-2. Victorian Partnership for Advance Computing, which is a part of the Australian Partnership for Advanced Computing Grid, hosts the centralised Grid Index Information Service (GIIS), a component MDS-2, while the remaining grid sites run the Grid Resource Information Service (GRIS) that connects to the Victorian Partnership for Advance Computing GIIS. A grid resource broker proposed by Venugopal et al. (2006), to access the APAC Grid has to contact the Victorian Partnership for Advance Computing GIIS, as contacting one of the other grid sites running a GRIS would only allow access to the information about that particular resource.

This isolation in resource information organisation in grids and among grids leads to the resource fragmentation problem. In this case, grid users get access to only a small pool of resources. Further, the institutions hosting the root GIIS service are central point of contact for the overall system. Failure of the root GIIS can partition the system, and can lead to significant performance bottlenecks. To overcome the limitations of centralised and hierarchical information services, proposed a decentralised grid resource information service based on a spatial index. Foster et al. (2003) utilised a Distributed Hash Table (DHT) routing substrate for delegation of d-dimensional service messages. DHTs have been proven to be scalable, self-organising, robust and fault-tolerant.
In et al. (2011) proposed a grid resource discovery service which organises data by maintaining a logical d-dimensional publish/subscribe index over a network of distributed grid brokers/grid sites. The spatial nature of the publish/subscribe index has the capability to respond to the complex grid resource queries, such as range queries involving various attribute types, including those that have a spatial component.

Further, the resource discovery system is extended to provide a abstraction/facility of a Peer-to-Peer (P2P) tuple space for realising a decentralised coordination network, which is explained by Buyya et al. (2001). The P2P tuple space can transparently support a decentralised coordination network for distributed brokering services. It provides a global virtual shared space that can be concurrently and associatively accessed by all participants in the system and the access is independent of the actual physical or topological proximity of the tuples or hosts. The grid peers maintaining the tuple space undertake the activity, related to job load-balancing across the grid-federation resources.

In grid computing, the network status and the resources status are to be managed effectively. If the network status or resource status is not in feasible level, then the total computation time will be increased dramatically. In grid computing, the user will encounter thousands of computers to utilize effectively and efficiently. The grid architectures serving as a middleware technology for various purposes like resource allocation management, job scheduling, data management, security and authorization.

Programming in grid computing involves more complexities which not only require a single-machine application instead it has to look the entire network. Some of the additional aspects in the grid computing shown in Myer (2003) are organized as dividing and combining data and results, data security, application security, testing, and redundancy and capacity plan. In
the mid-1990s, the grid metaphor was applied to computing, by extending and advancing the 1960s concept of “computer time sharing.” The grid metaphor strongly illustrates the relation to, and the dependency on, a highly interconnected networking infrastructure.

According to IBM, based on the white paper by Brown (2003), a grid is a collection of distributed computing resources available over a local or wide area network that appears to an end user or application as a large virtual computing system. The vision is to create virtual dynamic organizations through secure, co-ordinated resource sharing among individuals, institutions and resources. Grid computing is an approach of distributed computing that spans not only locations but also organizations, machine architectures, and software boundaries to provide unlimited power, collaboration, and information access to everyone connected to a grid.

1.4 KEY ISSUES IN GRID COMPUTING

The key issues in grid computing are:

- Components of grid computing systems/architectures
- Portal/user interfaces
- User security
- Broker function
- Scheduler function
- Data management function
- Job management and resource management
- Security considerations
- Grid deployment/management issues
- Challenges and approaches
- Availability of products by categories
- Business grid types
- Deploying a basic computing grid
- Deploying more complex computing grid
- Grid operation
- Economics of grid systems
- The chargeable grid service
- The grid payment system
- Communication and networking infrastructure
- Communication systems for local grids
- Communication systems for national grids
- Communication systems for global grids
- Metering service
- Accounting service
- Billing and payment service
- Grid system deployment issues and approaches
- Generic implementations
- Benefits of grid and status of technology
- Motivations for considering computational grids
- Readiness of grid computing for prime time
- Early suppliers and vendors challenges
- Future directions
- Stable standards supporting grid computing
- Virtual organization creation and management
- Service groups and discovery services
- Choreography, orchestration, and workflow
- Transactions

From the key issues, the virtualization analysed by Stillwell et al. (2012), is the most important requirement of grid systems, and the virtualization can span the following domains:

1. Server virtualization for horizontally and vertically scaled server environments. Server virtualization enables optimized utilization, improved service levels, and reduced management overhead.

2. Network virtualization, enabled by intelligent routers, switches, and other networking elements supporting virtual LANs. Virtualized networks are more secure and able to support unforeseen spikes in customer and user demand.

3. Storage virtualization (server, network, and array-based) technologies improve the utilization of current storage sub-systems, reduce the administrative costs, and protect vital data in a secure and automated fashion.

4. Application virtualization enables programs and services to be executed on multiple systems simultaneously. This computing approach is related to horizontal scaling, clusters, and grid computing, in which a single application is able to cooperatively execute on a number of servers concurrently.
5. Data center virtualization, whereby groups of servers, storage, and network resources can be provisioned or reallocated on the fly to meet the needs of a new IT service or to handle dynamically changing workloads.

1.5 RESEARCH ISSUES IN GRID COMPUTING

From the various generalized issues shown in section 1.3, major research issues in computational science are shortlisted and shown in the following:

- Scheduling
- Grid architecture models
- Grid topologies
- Reliability
- Resource balancing
- Security

1.5.1 Scheduling

The purpose of task scheduling in grid computing is to balance the load of entire grid system in such a way as to complete all the assigned workload as soon as possible and feasible. It is impossible for anyone to manually assign these loads in the large computing resources of grid system. As the environment status of grid architecture is changing frequently, the traditional job scheduling algorithm such as ‘First Come First Serve’ (FCFS), ‘Shortest Job First’ (SJF), etc., may not be suitable for a dynamic environment in grids. Therefore, job scheduling in the grid environment is an important issue. This research proposes an efficient job scheduling algorithm for solving these problems in the grid environment.
As the grid concept is growing in the modern era, it attracts a large number of researchers. There are a variety of research activities identified in the grid environment, in which the task scheduling and load balancing are the major research issues even today. Many scheduling algorithms for grid task are proposed in recent years; for example, load balancing by Moges et al. (2009) and the job scheduling by Kacsuk et al. (2007), in which the scheduling is classified in a variety of ways. The job scheduling method is composed of the following major steps:

- Submitting a new job to the system.
- Based on the predicted finishing time, the Resource Broker selects the “best” Computing Element for its execution. If the chosen Computing Element does not provide enough disk space for the input files, they are replicated.
- Agents are sent to every source Storage Element, and one to every destination node.

The agents will run on the nodes in the background as daemons (the environment for this will be assured by the agent hosts) and copy the files prior to the execution queue reaching the job requiring them.

1.5.2 Grid architecture models

There are different types of grid architectures to fit different types of business problems. Some grids are designed to take advantage of extra processing resources, whereas few grid architectures are designed to support collaboration between various organizations.
Grid type is selected primarily based on the business problem that is being solved. Taking the goals of the business into consideration will help the user to choose a proper type of grid framework. A business that wants to tap into unused resources for calculating risk analysis within their corporate data centre will have a different design than a company that wants to open their distributed network to create a federated database with few of their main suppliers. Such types of grid applications will require proportionately different designs, based on their respective unique requirements.

The selection of a specific grid type will have a direct impact on the grid solution design. Additionally, it should be mentioned that grid technologies are still evolving and tactical modifications to grid reference architecture is required to satisfy a particular business requirement.

### 1.5.3 Grid topologies

A topology view covers the following spectrum of grids:

- **Intra grids**
  - Intra grids are applied to single organizations
  - No partner integration
  - Single cluster

- **Extra grids**
  - Extra grids are applied to multiple organizations
  - Partner integration
  - Multiple clusters

- **Inter grids**
  - Inter grids are applied to multiple organizations,
  - Multiple partners
The simplest of these three topologies is the intra grid, which comprises merely a basic set of grid services within a single organization. The complexity of the grid design explained by Caminero et al. (2011) is proportionate to a number of organizations that the grid is designed to support, the geographical parameters and constraints. As more organizations join the grid, the non-functional or operational requirements for security, directory services, availability, and performance become more complex.

As more organizations require to access grid resources, the requirements for increased application layer security, directory services integration, higher availability, and capacity become more complicated. The philosophy behind grid computing is transparent, secure, and coordinated resource sharing and problem solving in dynamic, multi-institutional organizations. The resource sharing alluded to is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly protected, with resource providers and consumers defining clearly and carefully just what is shared, who is shared, and the conditions under which sharing occurs.

1.5.4 Reliability

High-end conventional computing systems use expensive hardware to increase reliability. They are built using chips with redundant circuits that vote on results, and contain much logic to achieve graceful recovery from an assortment of hardware failures. The machines also use duplicate processors with hot connectivity so that when they fail, that can be replaced without turning the other off. The systems are operated on special power sources that can start generators if utility power is interrupted. All of this builds a reliable system as elaborated in Dai et al. (2007), but at a high cost, due to the duplication of high-reliability components.
In future, there may be a complementary reliable approach that relies on software and hardware. A grid is just the beginning of such technology. More details are found in Shah et al. (2007) and Dobber et al. (2009). The systems in a grid can be relatively inexpensive and geographically dispersed. Thus, if there is a power or other kind of failure at one location, the other parts of the grid are not likely to be affected. Grid management software can automatically resubmit jobs to other machines on the grid when a failure is detected. In critical, real-time situations, multiple copies of the important jobs can be run on different machines throughout the grid.

Their results can be checked for any kind of inconsistency such as computer failures, data corruption and tampering. Such grid systems will utilize “autonomic computing” (Bahrami et al., 2010). This is a type of software that automatically heals problems in the grid, perhaps even before an operator or manager is aware of them. In the grid based industry, the principle explained in Murugesan (2008) is maintained in such a way that most of the reliability attributes achieved using hardware in today’s high availability systems can be achieved using software in a grid setting in the future.

1.5.5 Resource balancing

A grid federates a large number of resources contributed by individual machines into a virtual resource, as described by Kesavan et al. (2008). For applications that are grid-enabled, the grid can offer a resource balancing effect by scheduling grid jobs on machines with low utilization. This feature can prove invaluable for handling occasional peak loads of activity in larger organization which can happen in two ways:

- An unexpected peak can be routed to relatively idle machines in the grid.
If the grid is already fully utilized, the lowest priority work being performed on the grid can be temporarily suspended or even cancelled and performed again later to make room for the higher priority work.

Without a grid infrastructure, such balancing decisions are difficult to prioritize and execute. Occasionally, a project may suddenly rise in importance with a specific deadline. A grid cannot perform a miracle and achieve a deadline when it is already too close. However, if the size of the job is known, if it is a kind of job that can be sufficiently split into sub-jobs, and if enough resources are available after pre-empting lower priority work, a grid can bring a very large amount of processing power to solve the problem. In such situations, a grid can, with some planning, succeed in meeting a surprise deadline.

Other suitable benefits can occur using a grid for load balancing. When jobs communicate with each other, the Internet, or with storage resources, an advanced scheduler could schedule them to minimize communications traffic or minimize the distance of the communications. This can potentially reduce communication and other forms of contention in the grid.

Finally, a grid provides excellent infrastructure for brokering resources. Individual resources can be profiled to determine their availability and their capacity, and this can be factored into scheduling on the grid. Different organizations participating in the grid can build up grid credits and use them at times when they need additional resources. This can form a basis for grid accounting and the ability to fairly distribute the work on the grid.
1.5.6 Security

Security requires three fundamental services: authentication, authorization and encryption. A grid resource must be authenticated before any checks can be done as to whether or not any requested access or operation is allowed within the grid. Once the grid resources have been authenticated within the grid, the grid user can be granted certain rights to access a grid resource. This, however, does not prevent data in transit between grid resources from being captured, spoofed, or altered. The security service to insure that this does not happen.

Song et al. (2006) proposed secured grid scheduling, in which, the system failures, delays caused by infected hardware, software vulnerability, and distrusted security policy are taken as risk factor. In order to avoid such risks, the authors formulated a three risk-resilient strategies such as pre-emptive, replication, and delay-tolerant. Tinedo et al. (2011) combined DHT with secured models of Song et al. (2006).

The word security has its own set of terminology. The International Organization for Standardization (ISO) has defined the common security services found in modern IT systems. The list was first put in ISO 7498-2 (OSI Security Architecture) and later updated in ISO 10181 (OSI Security Frameworks). The components of grid security systems are listed by Jose and Seenivasagam (2011), which are shown below:

- Authentication - Authentication is the process of verifying the validity of a claimed individual and identifying the customers. It is not limited to human beings; services, applications, and other entities may require to be authenticated also.
• **Access control** – Assurance that each user or computer that uses the service is permitted to do what the user asks for. The process of authorization is often used as a synonym for access control, but it also includes granting the access or rights to perform some actions based on access rights.

• **Data integrity** – Data integrity assures that the data are not altered or destroyed in an unauthorized manner.

• **Data confidentiality** – Sensitive information must not be revealed to parties that it is not meant for. Data confidentiality is also often referred to as privacy.

• **Key management** – Key management deals with the secure generation, distribution, authentication, and storage of keys used in cryptography.

The Grid Security Infrastructure and a Public Key Infrastructure provide the technical framework (including protocols, services, and standards) to support grid computing with five security capabilities: user authentication, data confidentiality, data integrity, non-repudiation, and key management.

### 1.6 MOTIVATION

Tamilnadu state, a big state in southern India, has a population of around eight crores, in which around eight lakh students are writing higher secondary examination annually. After their higher secondary, around two lakh students prefer to study engineering every year. The engineering colleges in the entire state are affiliated to Anna University, Chennai. Therefore, a common counselling for admission to undergraduate and postgraduate courses is conducted by Anna University Chennai.
As the counselling is conducted at one place in the state capital Chennai, students have to travel almost 800 km with travel time of 16 hours and an average of 450 km with travel time of 10 hours. As the counselling is conducted at one place, every day around 3000-6000 students are forced to assemble Anna University Chennai Campus, which leads to an uncomfortable state. Therefore, this work proposes that the engineering counselling may be distributed in many places around the state. This will reduce long travel. This is possibly implemented using grid computing environment. In order to improve the performance of the proposed system, the six research issues of computational science which are discussed in section 1.4 are implemented in this thesis.

1.7 ORGANIZATION OF THESIS

This thesis is organized into seven chapters as described below:

Chapter 1 introduces proposed research area, discusses the research issues in the proposed domain, and describes the research focus of the proposed work.

Chapter 2 offers a brief explanation and review of the proposed research area by various researchers and implementations with discussions on their limitations.

Chapter 3 explains the grid architecture briefly and describes the overall design of the proposed work, the detailed architecture of the proposed grid architecture, which is termed interlinked mesh grid architecture with outer connection.
Chapter 4 describes the design of mathematical model and functionalities of the proposed grid control system with its importance, requirements and performance.

Chapter 5 deals with the mathematical model, definition and implementation of the proposed Hybrid Hash Table (HHT) and the performance analysis of proposed work and comparison with existing work.

Chapter 6 discusses the mathematical model, definition, improvements and implementation of proposed Parameter Optimized Ant Colony Optimization (POACO) and the performance comparisons with existing methodologies.

Chapter 7 sums up this thesis and highlights the performance of the proposed work. Further, this chapter also suggests the directions for future research in this area.