CHAPTER 1

Introduction

This chapter provides introduction to Parallel Computing, Distributed Computing and Grid Computing. It briefly discusses about types of Grids, Grid characteristics and key components of Grid computing. It presents the motivation and objectives of the research work followed by summary of key contributions. It ends with a discussion on the organization of the rest of this dissertation.

1.1 Parallel Computing

Traditionally, software has been written for serial computation, to be executed by a single computer having a single Central Processing Unit (CPU) and problems are solved by a series of instructions, executed one after the other by the CPU. Only one instruction executes at any moment in time.

Parallel computing is an evolution of serial computing that attempts to emulate what has always been the state of affairs in the natural world. It addresses many complex, interrelated events happening at the same time, in a sequential fashion. Some examples are:

- Planetary and galactic orbits
- Weather and ocean patterns

In the simplest sense, parallel computing is the simultaneous use of multiple resources to solve a computational problem. The resources can include, a single computer with multiple processors. There are two primary reasons for using parallel computing:

- Save time – wall clock time
- Solve larger problems
Other reasons might include:

- **Taking advantage of non-local resources** – using available compute resources on a wide area network, or even the Internet when local compute resources are not sufficient for a computation.
- **Cost savings** – using multiple “cheap” computing resources instead of paying more on a super computer.
- **Overcoming memory constraints** – isolated computers have very finite memory resources. For large problems, using the memory available with multiple computers may overcome this obstacle.

However, multiple tasks cannot run through parallel computing, for which distributed computing has evolved.

### 1.2 Distributed Computing

Distributed Computing is a type of computing in which different components and objects comprising an application can be located on different computers connected to a network. For example, a word processing application might consist of an editor component on one computer, a spell-checker application on a second computer, and a thesaurus on a third computer. In some distributed computing systems, each of the three computers could even be running a different operating system.

One of the requirements of distributed computing is a set of standards that specify how objects communicate with one another. There are currently two distributed computing standards: CORBA and DCOM [Distributed Computing].

Various distributed technologies are:

- Distributed Computing systems
- World Wide Web
- Application and Storage Service Providers
- Peer-to-Peer Computing systems
• Cluster Computing

The difference between parallel computing and distributed computing is, parallel computing brings many CPUs together in one box, and whereas distributed computing refers to the linking up of many individual computers to perform the task.

1.3 Problems with Parallel Computing & Distributing Computing

Problems with traditional parallel computing are:

• Expensive software
• Very high starting cost
• High maintenance cost
• Costly to upgrade

Because of these the computing power of an organization is generally not enough to finish its task in a reasonable amount of time.

Problems with various distributed technologies are:

• Cluster computing is local to domain and not amenable to resource sharing among participant from different domain.
• Lack of security features
• Respecting the policies of individual organizations participating in resource sharing.

Grid Computing addresses the above problems by having:

• **Resource sharing:** Resources in a grid belong to many different organizations that allow other organizations (i.e. users) to access them. Nonlocal resources can be used by applications, promoting efficiency and reducing costs of the resources while allowing its scalability.
• **Multiple administrations:** Each organization may establish different security and administrative policies under which their resources can be accessed and used.

### 1.4 Grid Computing

Grid computing is a form of distributed computing; the pioneers in this field are Dr. Ian Foster of Argonne National Laboratory and the University of Chicago, and Dr. Carl Kesselman of the Information Sciences Institute and the University of Southern California. In their 1998 seminal work, *The Grid: Blueprint for a New Computing Infrastructure*, they defined Grid computing as follows:

"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." [Ian Foster, Carl Kesselman, 1999]

Grid computing is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. The grid is a new approach to high-performance and large scale networked computing that has the potential to alter how computing resources are allocated for large projects. Built on the Internet and the World Wide Web, it is a new class of infrastructure comprising a set of high-speed computers, storage systems and networks, plus a set of grid services (or middleware) to coordinate the ensemble of resources. By providing scalable, secure, high-performance mechanisms for discovering and negotiating access to remote resources, the Grid promises to make it possible for scientific collaborations to share resources on an unprecedented scale, and for geographically distributed groups to work together in ways that were previously impossible [Arun Agarwal, 2004].

Grid computing area has attracted a number of researches in recent years due to the unique ability of marshalling collections of heterogeneous computers and resources, enabling easy access to diverse resources and services that could not be possible without a grid model.
A grid in reference to grid computing refers to loose resources and users that may be geographically separated and may be under different administrative domains. A grid uses the resources of many separate computers connected by a network (usually the Internet) to solve large scale computation problems and use most of idle time on many resources throughout the world. Such arrangements permit handling of data that would otherwise require the power of expensive supercomputers or would have been impossible to analyze.

1.4.1 Types of Grids

From an application perspective, grids are categorized into two types: compute grids and data grids. Although from a topology perspective, it can be argued that there are additional types that include clusters, intra-grids, extra-grids, and inter-grids. In reality, clusters, intra-grids, extra-grids, and inter-grids are better defined as stages of evolution that fall under compute grids, data grids, or a combination of both. The majority of the early grid deployments have focused on enhancing computation, but as data grids provide easier access to large, shared data sets, data grids are becoming increasingly important.

A compute grid is essentially a collection of distributed computing resources, within or across locations that are aggregated to act as a unified processing resource or virtual supercomputer. These compute resources can be either within or between administrative domains. Collecting these resources into a unified pool involves coordinated usage policies, job scheduling, queuing characteristics, grid-wide security, and user authentication. The benefit is faster, more efficient processing of compute-intensive jobs, while utilizing existing resources. Compute grids also eliminate the drawback of tightly binding specific machines to specific jobs, by allowing the aggregated pool to most efficiently service sequential or parallel jobs with fine-grained user attributes.

A data grid provides wide area, secure access to data. Data grids enable users and applications to manage and efficiently use database information from distributed locations. Similar to compute grids, data grids also rely on software for secure access and usage policies. Data grids can be deployed within one administrative domain or across multiple domains. In these cases the grid software and policy management
becomes critical. Data grids eliminate the need to unnecessarily move, replicate, or centralize data, translating into cost savings. Initial data grids are being constructed today, primarily serving collaborative research communities. Software vendors and large enterprises are currently investigating data grid solutions and services for business applications.

1.4.2 Grid Characteristics

Ten definitions extracted from main grid literature sources have been examined to find out the essential characteristics that a grid is supposed to have in order to be of any practical use.

1. Large scale: a grid must be able to deal with a number of resources ranging from just a few to millions. This raises the very serious problem of avoiding potential performance degradation as the grid size increases.

2. Geographical distribution: grid's resources may be located at distant places.

3. Heterogeneity: a grid hosts both software and hardware resources that can be varied ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, supercomputers and networks.

4. Resource sharing: resources in a grid belong to many different organizations that allow other organizations (i.e. users) to access them. Non-local resources can thus be used by applications, promoting efficiency and reducing costs of the resources while allowing its scalability.

5. Pervasive access: the grid must grant access to available resources by adapting to a dynamic environment in which resource failure is commonplace. This does not imply that resources are universally available but that the grid must tailor its behavior as to extract the maximum performance from the available resources.

6. Multiple administrations: each organization may establish different security and administrative policies under which their resources can be accessed and used. As a
result, the already challenging network security problem is complicated even more with the need of taking into account all different policies.

7. **Resource coordination**: resources in a grid must be coordinated in order to provide aggregated computing capabilities.

8. **Transparent access**: a grid should be seen as a single virtual computer.

9. **Dependable access**: a grid must assure the delivery of services under established Quality of Service (QoS) requirements. The need for dependable service is fundamental since users require assurances that they will receive predictable, sustained and often high levels of performance.

10. **Consistent access**: a grid must be built with standard services, protocols and interfaces thus hiding the heterogeneity.

### 1.4.3 Key Components of Grid Computing

Grid Computing contains the following major components [IBM developerWorks]:

- **Security**: Computers on a grid are networked and running applications. They can also be handling sensitive or extremely valuable data, so the security component of grid computing is of paramount concern. This component includes elements such as encryption, authentication, and authorization.

- **User Interface**: Accessing information on the grid is also quite important, and the user interface component handles this task for the user. It often comes in one of two ways:
  - An interface provided by an application that the user is running
  - An interface provided by the grid administrator, much like a Web portal that provides access to the applications and resources available on the grid in a single virtual space.
• **Workload management:** Applications that a user wants to run on a grid must be aware of the resources available. This is where a workload management service comes in handy. An application can communicate with the workload manager to discover the available resources and their status.

• **Scheduler:** A scheduler is needed to locate the computers on which to run an application and to assign the jobs required. This can be as simple as taking the next available resources, but this task often involves prioritizing job queues, managing the load, finding workarounds when encountering reserved resources, and monitoring progress.

• **Data Management:** If an application is running on a system that doesn’t hold the data the application needs, a secure, reliable data management facility takes care of moving that data to the right place across various machines, encountering various protocols.

• **Resource Management:** To handle core tasks such as discovering resources, monitoring resource status, launching jobs with specific resources, monitoring the status of those jobs, and retrieving results, a resource management facility is necessary.

### 1.5 Motivation

Why should we be interested in **Resource Management over a GRID**? A resource on a grid could be any entity that provides access to a service. This could range from compute servers to databases, scientific instruments and applications. In a heterogeneous environment like a grid, resources are generally owned by different people, communities or organizations with varied administrative policies, and capabilities. Naturally, obtaining and managing access to these resources is not a simple task.

The aim of resource management is to simplify the process of handling the resources used by users to complete their task. In the grid computing environment resource management is a collection of software components that allow users to access
heterogeneous resources transparently, without having to worry about availability, access methods, security issues and other policies.

1.6 Objectives

In a grid computing environment, as resources are widely dispersed, there is a possibility that resources can go down at any time. Dynamic resource addition, deletion and modification are possible. There are separate administrative policies across organizations in managing resources. We need to address the following questions, to use grid resources:

- How to know where the resources are?
- How to identify resources?
- How to get permission to use them?
- How to know what are the applications available in these resources?
- How to use the resources?
- How to submit remote jobs?
- How to get access to resources to all the machines simultaneously?
- What happens if a resource fails?
- How input/output files are managed?

To address these questions an efficient resource management system is required. Current resource management systems in grid computing, suffer from several limitations. They need a dedicated resource management monitoring station, which must be on a specific type of platform. Access to resource management system remote locations is accomplished by using adhoc schemes such as X-host applications in Linux based resource management system. This permits export of the presentation and running the management system from a remote workstation. However, the remote workstation also needs to be a Linux system.

The information provided by the existing systems is required but not sufficient because they lack in the topology information and integrated way of looking at various tasks involved in management of grid resources. These problems are addressed in our
Grid Management Information Server (GMIS) framework. In addition, questions on communication issues, computation issues, performance issues, subscription to the service data, etc., are also addressed in this framework.

The main objective of this research work is providing an efficient resource management framework for Discovery, Monitoring and Brokerage of resources in grid computing environment. Discovery, Monitoring and Brokerage are vital functions in a grid computing environment, particularly when that system spans multiple locations, as in that context no one is likely to have detailed knowledge of all components. Discovery allows us to identify resources or services with desired properties, while Monitoring allows us to detect and diagnose many problems that can arise in such contexts. Brokerage allows users to submit jobs over grid. These tasks require the ability to collect information from multiple, perhaps distributed information sources.

To summarize, Resource Discovery, Monitoring and Brokerage in grid are required for the following main reasons:

- To get the information about resources health status
- To construct and execute applications
- To describe grid resources
- For resource scheduling, allocation and usage

Following are the important design considerations made in developing GMIS framework.

- **Seamless and transparent user access to resource information** – All information related to resources in a grid can be accessed by the user by selecting options in the viewer without any need of understanding of the grid infrastructure.
- **Easy to use web-based interface** – The viewer is web based and platform independent. User can access the viewer from any operating system and web browser. The information and controls available in the viewer are presented to the user in an intuitive way, so that the user can perform all his activities with ease.
• **Collation of information from multiple resources** – The information of resources presented under various grids or managed by different GMIS can be viewed by the user within a single Viewer.

• **Efficient resource management** - In order to gain maximum benefit from resources, the GMIS provides appropriate resources based on the job requirements. GMIS maintains latest resource information along with their status.

• **Scalable and Adaptable framework** - GMIS doesn’t impose any kind of limitations on the grid or jobs submitted by the user. If large number of resources/jobs is to be managed then only hardware needs to be upgraded. GMIS can be integrated with other grid technologies like Globus by agreeing on data exchange formats.

• **Poll and Event based information gathering mechanisms** – The information about resources will be gathered using poll mechanism where GMIS send a request to resource agent. In case of events mechanism, the resources agents automatically send information to GMIS whenever there is change in resource status.

### 1.7 Contributions

The GMIS framework is developed as part of development of Grid Middleware for Integration of Dispersed Resources at University of Hyderabad (UH), Hyderabad. We claim that this is a parallel effort along with peers who are working in this domain. In this development already a scheduler [Pramod Kumar, 2006] is developed which can be used in GMIS framework to submit jobs.

Currently UH Grid group is working on various research areas of grid computing. Some of them are Resource Management in Grid Computing environment, Schedulers, High Availability Grids, High Capacity High Latency Transport Protocols for Grid Computing, Geo Science applications on Grid and Heterogeneous Distributed Shared Memory Computing.

Recent contributions made by this group are, developed an algorithm for high availability of data in grids [Chidambaram, 2008], Extending OGSA DQP framework for
Following is the summary of main contributions of the thesis:

1. Proposed distributed framework architecture called Grid Management Information Server (GMIS) for resource management in grid environment, which is a robust integrated comprehensive framework, compared to available frameworks.

2. Identified key requirements and design considerations that GMIS framework needs to support.

3. Compared and contrasted GMIS framework with the traditional resource management frameworks.

4. Developed a scalable and adaptable GMIS framework which can manage resources that are distributed over a Wide Area Network (WAN). One instance of GMIS can communicate with another instance of GMIS so that the user can get a seamless view of available distributed resources over the WAN.

5. Designed resource discovery algorithms with three different strategies i.e., dynamic, event based and user based resource discovery.

6. Designed topology manager service, for maintaining resource information as well as support for virtual clustering where users can group the resources anywhere as a single logical/virtual cluster.

7. Developed monitoring and data collection manager services to provide users with wide range of options to select resources based on their capabilities and historical performance.

8. Developed push and pull based algorithms for gathering resource information, so that grid user will have latest status about the resources as well as submitted jobs.
9. GMIS provides support for activation graph creation based job submission and execution i.e., submitted jobs can be divided into multiple tasks and results can be analyzed at each task level.

10. GMIS is developed in such a way that to provide all necessary infrastructure services and environment for supporting MDO framework.

11. Developed agent software that resides on each resource to respond to the GMIS requests and to generate events on behalf of resources.

12. Developed a single intuitive web based interface to provide access to all GMIS services in an integrated fashion.

1.8 Thesis Organization

Chapter 2 presents overview of Grid computing and extensive literature survey on basic grid technologies and current grid activities. It discusses about resource management and some of the available resource management frameworks in detail. It gives a comparative study of GMIS framework with other resource management frameworks.

Chapter 3 proposes details of the framework called Grid Management Information Server (GMIS) architecture for resource management in grid environment for effective management of distributed resources. This framework architecture addresses the resource management requirements like Discovering, Monitoring and Brokerage of resources in grid environment.

Chapter 4 emphasizes on the design aspects of GMIS framework with data flow diagrams and data models. Algorithms for various tasks like discovery, polling and parsing, event handling are discussed in detail. It discusses how job execution can be done using GMIS framework. Message passing among GMIS components and communication between resources’ agent software and GMIS through communicator server is also discussed in this chapter.

Chapter 5 discusses about GMIS processes and interface modeling details of GMIS framework along with Graphical User Interfaces. It also discusses about testing environment for GMIS. Some experiments and results are presented in this chapter.
Chapter 6 presents case studies of GMIS framework. Multidisciplinary Design Optimization (MDO) application is being looked at as a case study for the GMIS framework. This chapter gives literature survey on MDO, UH-MDO framework over grid and then shows how MDO framework can use GMIS framework. GMIS framework can be applied for virtual clustering. This also presented in this chapter.

Chapter 7 concludes the thesis and proposes potential future work areas.