CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO GRID

A grid is an open system, a large collection of autonomous systems giving individual users the image of a single virtual machine with a rich set of hardware and software resources. Grid computing involves high performance parallel machines and clusters that provide major resources for simulation, analysis and other compute intensive activities. Computational Grid is a collection of distributed, heterogeneous resources which can be used as an ensemble to execute large-scale applications.

The common goal of grid is to create an illusion of a large and powerful self managing environment out of a large collection of connected heterogeneous systems sharing various combinations of resources. The shared resources and users typically span different organizations. The Grid infrastructure provides the ability to dynamically link together shared resources as an ensemble to support the execution of large scale, resource intensive, and distributed applications. Grid applications often involve large amounts of data and computing, and often require secure resource sharing across organizational boundaries. Although a grid can be dedicated to a specialized application, it is
more common that a single grid will be used for a variety of different purposes. Grids are constructed with the aid of general-purpose grid software libraries known as middleware. Grid middleware is a specific software product, which enables the sharing of heterogeneous resources and Virtual Organizations. It is installed and integrated into the existing infrastructure of the involved companies and provides a special layer placed among the heterogeneous infrastructure and the specific user applications. Major grid middlewares are Globus toolkit, gLite and UNICORE (Foster I 1997).

1.1.1 History

The term grid computing was originated in the early 1990s as a metaphor for making computer power as easy to access as an electric power grid. CPU scavenging and volunteer computing were popularized in 1997 by distributed.net and later in 1999 to harness the power of networked PCs worldwide, in order to solve CPU-intensive research problems. The ideas of the grid (including those from distributed computing, object-oriented programming, and Web services) were brought together by Ian Foster, Carl Kesselman, and Steve Tuecke, widely regarded as the "fathers of the grid". They led the effort to create the Globus toolkit incorporating not just computation management but also storage management, security provisioning, data movement, monitoring, and a toolkit for developing additional services based on the same infrastructure, including agreement negotiation, notification mechanisms, trigger services, and information aggregation. While the Globus Toolkit remains the de facto standard for building grid solutions, a number of other tools have been built. In 2007 the term cloud computing came into popularity, which is conceptually similar to the canonical Foster definition of grid computing (in terms of computing resources being consumed as electricity from the power grid). Grid computing is often associated with the delivery of cloud computing systems as exemplified by the AppLogic system.
1.1.2 Types of Grids

A grid provides full-scale integration of heterogeneous computing resources of any type viz. processing units, storage units, communication units. Real-world grid implementations are more specialized and generally focus on the integration of certain types of resources. As a result, there are different types of grids, which are described below:

- **Computational grid** (Bart Jacob 2005): Computational grid is a grid that has the processing power as the main computing resource shared among its nodes. This is the most common type of grid and it has been used to perform high-performance computing to tackle processing-demanding tasks.

- **Data grid** (Bart Jacob 2005): Like computational grid, data grid has the processing power as the main computing resource shared among their nodes. A data grid has the data storage capacity as its main shared resource. Such a grid can be regarded as a massive data storage system built up from portions of a large number of storage devices.

- **Network grid** (Bart Jacob 2005): Network grid is also known as delivery grid. Such a grid has as its main purpose to provide fault-tolerant and high-performance communication services. In this sense, each grid node works as a data router between the two communication points, providing data-caching and other facilities to speed up the communications between such points.
There are two different categories of processes (Bart Jacob 2005):

1. Terminating process: A terminating process is considered as a process, that runs and exits (terminates). For these processes, the deadline is the time at which it completes all its task and the compute time is the CPU time it needs to complete its entire task.

2. Non-terminating process: A Non-terminating process is considered as a process for which the terminating time is not given much importance, but rather the time between events. Example: video, audio servers and editors. For these processes, the compute time is the CPU time that the process needs to compute its periodic event and the deadline is the time at which it must have the results ready. Non terminating processes are divided into two classes. They are

- Periodic: A periodic process has a fixed frequency at which it needs to run.
- Aperiodic: Aperiodic processes have no fixed, cyclical, frequency between events. Event interrupts will occur sporadically and event computation times will vary dramatically.

1.1.3 Design Considerations and Variations

Grids are formed from resources belonging to multiple individuals or organizations (known as multiple administrative domains). The disadvantage of this feature is that the computers which are actually performing the calculations might not be entirely trustworthy. The designers of the system must thus introduce measures to prevent malfunctions or malicious participants from producing false, misleading, or erroneous results, and from using the system as an attack vector. This often involves assigning work randomly to different nodes
(presumably with different owners) and checking that at least two different nodes report the same answer for a given work unit. Discrepancies would identify malfunctioning and malicious nodes. Due to the lack of central control over the hardware, there is no way to guarantee that nodes will not drop out of the network at random times. These variations can be accommodated by assigning large work units (thus reducing the need for continuous network connectivity) and reassigning work units when a given node fails to report its results in expected time.

The participating nodes must trust the central system that the access that has been granted will not be abused, by interfering with the operation of other programs, mangling stored information, transmitting private data, or creating new security holes. Public systems or those crossing administrative domains (including different departments in the same organization) often result in the need to run on heterogeneous systems, using different operating systems and hardware architectures. With many languages, there is a tradeoff between investment in software development and the number of platforms that can be supported. Cross-platform languages can reduce the need to make this trade off. The middleware is seen as a layer between the hardware and the software. On top of the middleware, a number of technical areas are considered, and these may or may not be middleware independent. Example areas include SLA management, Trust and Security, Virtual organization management, License Management, Portals and Data Management. Figure 1.1 shows the typical grid environment.
1.1.4 Open Grid Forum (OGF) (Bart Jacob 2005)

OGF is an open community committed to drive the rapid evolution and adoption of applied distributed computing. The OGF community includes leaders and practitioners drawn from academia, enterprises, vendors and government organizations. OGF is open to everyone who is willing to participate, to discuss trends, share experiences, solve problems, and develop standards that accelerate the adoption, use and development of applied distributed computing technologies and environments. OGF accomplishes its work through open forums that build the community, explore trends, share best practices and consolidate these best practices into standards.
1.2 IMPORTANCE OF GRID COMPUTING

In most organizations there are large amounts of underutilized computing resources. Most desktops are busy less than 5% of time. Often machines have enormous unused disk drive capacity. Grid computing provides a framework for exploiting these underutilized resources. Many industries and scientific communities require the use of parallel computing in order to run applications or solve certain problems. Grid computing provides a framework that allows jobs to be split up into multiple sub jobs and sub jobs are made to execute in parallel on different machines in Grid. The grid can help in enforcing security rules and implement policies, which can resolve priorities for both resources and users. Grid also offers load balancing. If jobs running on grid require a high level of communication between each other, they could be scheduled in a manner that minimizes the cost of communication or the amount of traffic on their communication line.

Grid is found to be important (Bart Jacob 2005) because

- Grid improves efficiency/reduces costs
- Grid exploits under-utilized resources
- Grid enables collaborations
- Grid has virtual resources and virtual organizations (VO)
- Grid increases capacity and productivity
- Grid supports heterogeneous systems
- Grid provides reliability/availability
1.3 REQUIREMENTS FOR GRID

Some of the vital requirements for grid (Bart Jacob 2005) are as follows

1.3.1 Hardware requirements

A grid environment is made up of computing resources. The basic hardware requirements that must be satisfied by the grid implementation are as follows:

- Every computing resource must have enough computing power and data storage capacity to properly run the grid platform.
- The computing resources do not need to be directly connected to each other.
- The resource needs to know about the entity that takes it to the grid; an entity could be an internal scheduler, or a data server.
- Computing resources can be indirectly connected through routers, gateways, hubs, switches, bridges, and wireless connections, by which a data packet can be sent from one computing resource to another. Depending on the type of grid, there are additional hardware requirements that have to be satisfied.

Computational grid

- The overall computing power of the grid is calculated as the sum of the computing power of its nodes. The efficiency of a grid will largely depend on the application that it executes.
- The overall performance of a grid also largely depends on the quality of the communication links that interconnect the nodes.
- The time spent on data exchange during the execution of an application should be negligible compared to the time spent on processing this data.
Data grid

- The overall data storage capacity of a grid is the sum of the storage capacity made available for the grid in all its nodes.
- Each node should have enough storage capacity to house the grid platform and to let the computer users perform their daily activities.

1.3.2 Software requirements

The basic software requirements that must be satisfied by any grid implementation are as follows:

- There must be interoperability among grid platforms of all the computing resources.
- Network software must be properly configured to allow the direct or indirect communication between any pair of computing resources. i.e., there must exist at least one logical path by which two computing resources can exchange data.
- The platform should be available through some sort of on-line network, such as the Internet or through commonly used storage medias such as CD-ROMs. The installation itself should be straightforward, requiring few and simple steps.
- Remotely and automatically upgrading the grid platform and the code for its applications
- Remotely monitoring the computing resource
1.4 THE LAYERED GRID ARCHITECTURE

The layered grid architecture consists of fabric layer, connectivity layer, resource layer, collective layer and application layer. The main focus of grid architecture is interoperability among service providers and users in order to establish the sharing relationships. The interoperability necessitates common protocol at each layer of the architectural model. This architectural model defines common mechanisms, schemas, interfaces at each layer. Each layer shares the behavior of the underlying component layers. The Layered Grid Architecture is shown in Figure 1.2.

![Figure 1.2 Layered Grid Architecture](image)

1.4.1 Fabric layer

Fabric layer defines the interface to local resources, which are shared. This includes computational resources, data storage, networks, catalogs, software modules and other system resources. In this layer, the diverse mix of resources like individual computers, condor pools, file systems, archives, metadata catalogs, networks, sensors etc… are shared.
1.4.2 Connectivity layer

The connectivity layer defines the basic communication and authentication protocols required for Grid specific networking-service transactions. It deals with security like uniform authentication, authorization, message protection and single sign–on, delegation, identity mapping, certificate authorities, certificate and key management.

1.4.3 Resource layer

The resource layer is responsible for allocating resources to jobs. The Grid Resource Allocation Management (GRAM) (Hoschek W 2000) controls remote allocation, reservation and monitoring. It has Grid FTP which is a high-performance, secure, reliable data transfer protocol optimized for high bandwidth wide-area networks. It also has Information Provider Component called Grid Resource Information Service (GRIS).

1.4.4 Collective layer

The collective layer has resource brokers which perform resource discovery and allocation. It has a meta directory service which supports dynamic resource collections. The collective layers are responsible for coherency control, replica selection, task management and virtual data catalog.

1.4.5 Application layer

The application layer is the highest layer of the layered Grid Architecture. This layer includes applications in Science, Engineering, Business, Finance and Development toolkits. This layer includes serviceware, which
performs general management functions like tracking the service providers and the users.

1.5 GRID APPLICATIONS

Grid technology has been applied to computationally intensive scientific, mathematical, and academic problems. Grid allows technical organizations to carry out large-scale modeling or computing by simultaneously using the resources at their sites. A computational grid provides high performance computing, a data grid provides large storage capacity and a network grid provides high throughput communication that is useful for a variety of applications such as virtual conferences and live multimedia conferencing. It is used as a back office data processing in support for e-commerce and Web services.

In education, grids play a major role in providing, an opportunity for a new style of collaborative learning, an aid to online posters and discussions with students at other schools, an easy way to present and review results, an easy way to conduct peer-to-peer discussions, a rapporteur of presentations and discussions and a single portal to distributed resources.

1.6 MATCHMAKING IN GRID

Matching is a common operation in many areas of computer science. In computer science, the term matching refers to a process of evaluation of the degree of similarity of two objects. Before resources can be allocated to run an application, a user must select resources appropriate to the requirements of the user. This process of selecting resources based on user requirements is called “resource matching”. Many matchmaking based technologies address the issue of
grid resource matching. The grid matchmaking process involves three types of agents: consumers (*users*), producers (service *providers*) and a matchmaking service. A matchmaking service uses a matching algorithm to evaluate a matching function which computes the matching degree. Users describe their applications with Classads (Vijay Subramani 2002) language and submit them to Matchmaker. Classads allows users to define custom attributes for jobs. The service provider registers resources and its capabilities with the matchmaker. The matchmaker then matches user’s requirements with the available resources’ capabilities.

### 1.7 CHALLENGES IN MATCHMAKING

The grid environment raises many challenges when trying to match the shared resources with the request. They are

- The large pool of resources increases the searching complexity of the matchmaking process which in turn reduces the speed of the matchmaking process.
- The resources in the resource pool are dynamic in nature. Therefore it is difficult to converge the accuracy on the matchmaking process. This reduces the number of successful jobs.
- There would be diversity in the Grid resource behavior which results in job failures.

### 1.8 MOTIVATION OF THE WORK

The extensive importance of matching the user’s requirements with the resource capabilities and the challenges in matchmaking motivates the researcher to focus on matchmaking techniques. Each matchmaking technique uses different methodologies to match the request with the resource. But the
matchmaking techniques failed to provide best allocation of resources. Therefore an attempt has been made to design efficient Three Dimensional Matchmaking Model (TDMM).

1.9 PROBLEM STATEMENT

The extensive literature survey in the field of matchmaking enabled to identify the problems that exist in the matchmaking and to define them precisely and clearly as follows:

- To speed up the allocation process by minimizing the searching complexity.
- To make the system more reliable to the changing environment by considering the dynamicity of the resources.
- To assign best resource to job by considering the diversity in the resource behavior.
- To assure the successful allocation of job by ranking the resources and rescheduling the ranked resources

1.10 CONTRIBUTION

In this dissertation, the following contributions were made

- The parameters for matchmaking process in grid are analyzed and Three Dimensional Parameters are proposed for the matchmaking process. They are static parameters, dynamic parameters and behavioral parameters.

- A novel Three Dimensional Matchmaking Model (TDMM) has been designed. Adequate experiments have been performed to analyze the performance of TDMM.
i) Analysis has been done to study the performance of the TDMM under various input load levels. The performances are studied against the metrics viz. average response time, number of successful jobs, number of failed jobs.

ii) Analysis has been done to study the performance of TDMM under various resource ingress and egress levels.

iii) Analysis has been done to study the performance of TDMM with job categorization and TDMM without job categorization.

iv) Analysis has been done to study the performance of TDMM with batch of ‘new’ jobs and a batch of ‘new’ and ‘mature’ jobs.

- An extensive mathematical analysis of TDMM has been carried out. The analysis of scalability has been carried out mathematically using scaled problem analysis for TDMM. Different hypothesis and lemmas have been provided to prove the efficient matchmaking process of TDMM.

- A novel Dual Queue Three Dimensional Matchmaking Model (DQTDMM) has been designed with two separate queues, one for “mature” jobs and another for “new” jobs. Adequate experiments have been performed to analyze the performance of DQTDMM.
  
i) Analysis has been done to study the performance of the DQTDMM under various input load levels. The performances are studied against the metrics viz. average response time and number of successful jobs.

ii) Analysis has been done to study the average response time of DQTDMM with TDMM.

iii) Analysis has been done to study the number of successful jobs of DQTDMM with TDMM.

- An extensive mathematical analysis of DQTDMM has been carried out to analyze the impact of dual queues in matchmaking in DQTDMM.
1.11 ORGANIZATION OF THE THESIS

The rest of the thesis is organized as follows: The chapter 2 provides an extensive literature survey of various matchmaking approaches and discussions on different matchmaking approaches are presented. The chapter 3 provides the description of three dimensional parameters. The chapter 4 explains the architectural design of novel Three Dimensional Matchmaking Model (TDMM). The chapter 5 provides the experimental analysis of the TDMM in which the performance of TDMM was evaluated using significant performance metrics. The chapter 6 provides the mathematical analysis of TDMM. The chapter 7 provides the design of Dual Queue Three Dimensional Matchmaking Model (DQTDMM). In addition mathematical analysis and experimental analysis has been done for DQTDMM. The chapter 8 provides the summary of the research work and the future directions.