Chapter 1
Introduction

This chapter introduces the context of the research to be presented in this thesis. It starts off with an introduction to the general area of Grid Networks and then discusses about the motivation and challenges for load balancing in such environments. Then, it presents an overview of various load balancing algorithms and presents the primary contributions of this research. The chapter ends with a discussion on the organization of the rest of the thesis.

1.1. Grid Networks
With the advances in hardware and software technologies there is an increased interest in exploiting large-scale parallel and distributed systems for real-time and large-scale applications. Grid computing a form of distributed system originated in the early 1990s. It enables, sharing, exchange, selection, and aggregation of heterogeneous computing and storage resources which can be accessed as a power grid that are spread geographically across multiple administrative domains.

It is a super virtual computer used to execute large scale applications such as DNA analysis, seismic analysis, volcano analysis, protein folding, genome sequencing, etc. Grid networks can be categorized as a Computational Grid,
Data Grid, or a Service Grid depending on whether they serve computationally-intensive applications, data-intensive applications, or applications requiring distributed services.

1.2. Load Balancing
Because of the possible differences in the processing capacities and uneven job arrival rates, the load on different computers in the grid can differ significantly. Consequently, it leads to poor system performance. One can improve the performance of such a system by an appropriate allocation of the workload among the computers, and is commonly referred to as job allocation or load balancing. Given a large number of jobs, load balancing distributes the jobs to the computers optimizing a given objective function.

1.2.1. Taxonomy of Load Balancing Algorithms
Casavant et al presented a hierarchical taxonomy [90] for load balancing in general purpose parallel and distributed computing systems, which is also appropriate for grid computing as grid is also a form of parallel and distributed systems. The taxonomy of load balancing is presented in Figure 1.1 [90].
**Local Vs Global Scheduling**

Local scheduling is performed by the operating system of a processor and consists of the assignment of processes to the time slices of the processor. On the other hand, global scheduling is concerned with the assignment of processes in a multi-processor system. Because grid network consists of multiple processors that are geographically distributed, grid scheduling falls under global scheduling.

![Classification of Load Balancing Algorithms](image-url)

**Figure 1.1. Classification of Load Balancing Algorithms**
**Static Vs Dynamic Algorithms**

Static load balancing algorithms assume that all information related to load balancing decisions that is, the characteristics specific to the jobs, the computing nodes, and the communication network are known beforehand. Load balancing decisions are made either deterministically or probabilistically at compile time and remain constant throughout the execution. However, the assumption that the characteristics of the jobs, the computing resources and communication network are all known beforehand and remain constant may not apply to a grid environment.

In contrast, dynamic load balancing redistributes the processes among the processors during execution time by transferring tasks from the heavily loaded processors to the lightly loaded processors with the goal of improving the performance of the applications. Usually a dynamic load balancing algorithm is described by three policies:

- **Information policy**, used to exchange state information (number of jobs waiting in the queue to be processed) between the processing elements.
Chapter 1: Introduction

- **Transfer policy**, which determines whether a job should be processed locally or transferred to a remote processor.

- **Location policy**, which identifies the processing element to which a job should be transferred.

Definitely, the static algorithm is simpler to implement and has less runtime overhead. However, dynamic approaches may lead to better performance than static approach. Moreover, dynamic algorithms need not be aware of the run-time behavior of the applications in advance. But, the major disadvantage of dynamic load balancing schemes is the run-time over head due to:

- The exchange of load information among processing elements,

- The decision-making process for choosing jobs and processors for transfers, and

- The communication delays incurred due to relocation.

Static load balancing algorithms are presented in [13], [32],[72], and dynamic load balancing algorithms are presented in [11],[29],[62].
Optimal Vs Sub-Optimal Algorithms

In the event that all information governing the state of the system, and resource requirements of all applications are known, an *optimal allocation* can be derived depending on some optimization criterion ([3],[87]) such as minimization of total process completion time, maximization of system throughput, and so on. If arriving at an optimal solution is computationally infeasible, suboptimal solutions to the load balancing problem may be tried [42]. Within the realm of *suboptimal allocation* to the load balancing problem, there are two general categories-approximate and heuristic allocations.

Approximate Vs Heuristic

Rather than searching the entire solution space for an optimal solution, an *approximate algorithm* finds a solution that is adequately good. This technique can significantly decrease the time taken to find an acceptable solution if some metric is available for assessing the solution. There are four different categories of load balancing algorithms which come under approximate algorithms.

1) Solution space enumeration and search [16]

2) Graph Theoretic [89]
3) Mathematical programming [49]

4) Queuing theoretic [84]

A Heuristic algorithm [42] finds the solution by state space search. At successive stages of a program, instead of generating all possible solution branches, a heuristic algorithm selects branches that are more likely to produce acceptable solutions for use in the next step of the program.

**Distributed Vs Centralized Algorithms**
The responsibility of load balancing may be relegated to central authority, or it may be distributed among the processing elements. For the smaller set of machines, centralized authority has the advantage of ease of implementation, but suffers from the lack of scalability and fault tolerance [21]. For making the load balancing scheme scalable one can organize the dispatchers into a tree structure [14], also called as hierarchical load balancing.

In decentralized systems, distributed schedulers interact with each other and do assignment of jobs to processing elements [91]. Because there is no central authority for job allocation, the state of all systems is not collected at a single point. Thus, the problems of communication bottleneck and single point failures present
in centralized systems are prevented in decentralized systems which make the system more scalable. Consequently, it provides better fault-tolerance and reliability than centralized systems without fall-back or high availability solution.

**Cooperative Vs Non Cooperative**

In *Cooperative load balancing* the distributed components cooperate with each other in order to achieve a common system-wide goal [53]. In contrast, in non-cooperative load balancing individual processors make decisions independent of the actions of the other processors. In fact, each processor acts selfishly in order to achieve its own goal independent of others ([47],[48]).

**1.2.2. Load Balancing a Data Grid**

Distributed data intensive applications generate large number of tasks/jobs that reference and generate datasets may be of the order of Giga bytes and higher, like for example in high energy physics [95] or life sciences [7]. Such large scale data sets need specialized storage and management services which are provided by Data Grids. Data grids are used to harness geographically distributed resources for data-intensive problems by providing remote
Chapter 1: Introduction

data set storage, access management, replication services [33], and data transfer protocols [93].

Data Grids deal mainly with data migration, that is, with the decisions concerning the place where application-related datasets should be moved to. The data migration cost depends on the storage resources, data transfer protocols, network topology, network conditions, and size of the dataset. The effects of data migration in Grids are considered in [26]. One of the most common data migration techniques is data replication, which distributes replicas of datasets across the sites. By increasing data availability, the process of data replication reduces access time and bandwidth consumption and improves the reliability and fault-tolerance of the Grid. When different sites hold replicas of a particular dataset, a decision is to be made in selecting the best replica among them, and is called replica selection process. The best replica is the one that optimizes a required performance criterion such as the access time, the cost and the security ([79],[45],[96]). To select a best replica, a predictive approach can be employed, which uses the history of data transfer times in predicting the current data transfer throughput [79].

Techniques that jointly deal with the task scheduling and data replication problems are proposed in
[5],[6],[26],[45]. Particularly, in [45], the task scheduling and the data replication problems are decoupled, while in [5] the problem is addressed by integrating scheduling and replication strategies.

1.2.3. Objective Functions
From the optimization perspective, the objective of the load balancing schemes can be to provide a system-optimal solution where the focus is on optimizing the goal of the entire system over all jobs, or individual-optimal solution where each job optimizes its own goal independent of other jobs.

Extensive studies exist on the load balancing that determines a load allocation that yields a system-wide optimal solution ([40],[52],[88]). A few studies exist on load balancing that provides individual-optimal solutions which are based on game theory ([47],[48]).

1.2.4. Performance Analysis
It is essential to analyze the performance of load balancing algorithms to determine the best algorithm. A number of metrics are used depending on the desired outcome of performance analysis. Mostly, load balancing algorithms attempt to minimize the total average task delay [100] and
maximize resource utilization. Other performance metrics considered are the average task slow down [91], which is the ratio of the task’s total delay to its actual run time, the probability to miss the deadline, and several other metrics. Some works incorporate economic models in load balancing a Grid [74] that take into account deadline and budget constraints. Another important metric in load balancing is fairness addressed in ([43],[65]).

1.3. Contributions of the Thesis
Extensive studies exist on static and dynamic load balancing in grid systems, that consider optimization of the entire system i.e., minimization of mean response time of the entire system. They do not consider each individual’s objective. As a result, there may be some jobs that experience larger response times than others. There are also studies on load balancing in grid systems that consider optimization of individual response times as their main objective. However, such load balancing schemes do not consider system optimality. In this study, a pricing mechanism using competitive equilibrium approach for load balancing is proposed which simultaneously and individually strives to
minimize response time of each job and mean response time of all jobs.

The contributions of thesis are summarized as follows:

1. This thesis discusses key concepts of game theory, competitive equilibrium theory, and their applications in computer science. The objective of this endeavor is to identify technologies and algorithms developed in related areas that can be applied to the target research area.

2. This thesis presents static load balancing of a grid using competitive equilibrium approach.

   A grid system consisting of heterogeneous computers connected by a communication network and shared by various users is considered. The jobs of various users arrive at centralized dispatcher with different arrival rates following an exponential distribution.

   The grid system considered is translated to Fisher’s market model, where users are buyers, and goods are computing resources. Then a pricing mechanism using competitive equilibrium theory is proposed which determines prices for computing
resources so that each user consumes the computing resources such that its objective is optimized, subject to its budget constraints and the market clears.

Competitive Equilibrium Scheme for load balancing is compared with two other load balancing schemes-Global Optimal Scheme and Nash Game Scheme.

3. This thesis presents competitive equilibrium approach for load balancing a computational grid with communication delays.

A grid system consisting of heterogeneous computers interconnected by a communication network, shared by various users is considered. The jobs of various users arrive at each computer with different arrival rates. Each computer determines whether a job should be processed locally or moved to another computer for execution. Load balancing is achieved by moving some jobs from heavily loaded computers to the lightly loaded computers. As a result, a communication delay is incurred due to moving of jobs for remote processing. Therefore, the overall response time of user job is sum of expected node delay at each computer and expected communication delay due to transfer of a job.
In this study Fisher’s market model is adopted for the grid system considered, and competitive equilibrium prices for the computing resources and allocation of jobs to the resources at these prices are determined. For the purpose of evaluation two other load balancing schemes—Global Optimal Scheme and Nash Equilibrium Scheme are studied.

4. This thesis presents recursive competitive equilibrium approach for dynamic load balancing a grid system.

Static load balancing scheme for grid system is extended to dynamically load balance a grid system. At first, the dynamic case of Fisher’s market model is defined, and then recursive competitive equilibrium properties are stated.

The dynamic model of grid system is translated to dynamic Fisher’s market model, and recursive competitive equilibrium is obtained. The proposed Recursive Competitive Equilibrium approach is evaluated and compared with static load balancing using Nash Equilibrium scheme and Competitive Equilibrium scheme.
5. This thesis presents competitive equilibrium approach for load balancing a data grid.

A grid system consisting of a set of computing resources and storage resources and an application composed of various jobs each accessing a set of datasets is considered. Datasets are replicated on various storage sites.

The problem of load balancing a data grid is divided into two parts—for each job data set replicas that the job will use are selected and the site where these datasets will consolidate for the job’s execution is decided.

In this study, three approaches for load balancing a data grid using Global Optimal Scheme, Nash Equilibrium Scheme, and Competitive Equilibrium Scheme are proposed and compared against each other.

1.4. Organization of the Thesis
This thesis is divided into eight chapters. The chapter wise outline of the thesis is as follows:

Chapter 2 of the thesis gives an introductory overview of game theory concepts and its applications to computer science.
Chapter 3 of the thesis discusses key concepts of competitive equilibrium theory and its applications to computer science.

Chapter 4 of the thesis presents static load balancing scheme for a grid system using competitive equilibrium approach.

Chapter 5 of the thesis proposes competitive equilibrium approach for static load balancing of a grid system taking into account communication delays.

Chapter 6 of the thesis implements dynamic load balancing of a grid system using recursive competitive equilibrium approach.

Chapter 7 of the thesis deals with load balancing of a data grid and presents three approaches—Global Optimal Scheme, Nash Equilibrium Scheme, and Competitive Equilibrium Scheme.

Chapter 8 of the thesis presents conclusions and the ideas for future work.