# Load Balancing and Job Migration in Grid Environment

Thesis submitted in partial fulfillment of the requirements for the award of degree of

**Master of Engineering**

in

**Software Engineering**


By:

**Paritosh Kumar**

(80731018)

Under the supervision of:

**Dr. Inderveer Chana**

Sr. Lecturer (SG)



COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004


**JULY 2009**

# Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Load Balancing and Job Migration in Grid Environment"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Inderveer Chana and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

***(Paritosh Kumar)***

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

03/07/09

**(Dr. Inderveer Chana)**

**Sr. Lecturer(SG)**

Computer Science & Engineering. Department

Thapar University, Patiala

Head

Computer Science & Engineering. Department

Thapar University, Patiala

**(R.K.SHARMA)** 9/7/09

Dean (Academic Affairs)

Thapar University, Patiala

# Acknowledgement

I wish to express my deep gratitude to Dr. Inderveer Chana, Senior Lecturer and Dr. Seema Bawa Professor & Head, Computer Science & Engineering Department for providing their uncanny guidance and support throughout the preparation of the thesis report.

I am also heartily thankful to Dr. Maninder Singh, Associate Professor, Computer Science and Engineering Department and Dr.V.P.Singh, Assistant Professor, Computer Science and Engineering Department for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members, T.U. Grid Group and all my friends especially Arindam Choudhury, Jarrar Alam Rana, Sandeep Khode , Darshan singh and Ms. Shashi (Research Scholar) who were always there at the need of the hour and provided all the help and support, which I required for the completion of the thesis.

Last but not the least; I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

**Paritosh Kumar**

**(80731018)**

# Abstract

Grid Computing is emerging as wide scale distributed infrastructure that promises to support resource sharing and synchronised problem solving in dynamic and hetrogeneous environment. Grid Computing is becoming a generic platform for high performance and distributed computing due to which it is being adopted in various areas like academic, industry, and research use. Grid Computing has progressed a lot, yet, there are some areas of concern, like resource management, resource scheduling, load balancing and security over which, research is still in progress.

Load balancing is a great challenge in dynamic and heterogeneous environment like Grids. Load balancing is a technique to enhance resources, utilizing parallelism, improve throughput, and to cut response time through an appropriate distribution of the applications. The main goal of load balancing is to provide a distributed, low cost, scheme that balances the load across all the processors. An effective and efficient load balancing algorithm is required to balance the load in Grid environment, which is a tedious task due to basic nature of Grid environment.

Focus of this thesis is to analyze all the existing load balancing algorithms and to do a comparative study. After considering all pros and cons of all the existing algorithms,a new load balancing algorithm has been proposed based on all positive aspects of existing algorithms.The proposed algorithm has been implemented and tested in real Grid environment establised at Research Lab CSED,TU. Along with this ,a job migration technique for balancing the load in Grid Environment has also been discussed.

# Table of Content

## List of Figures

## List of Tables

# Chapter 1 Introduction

Grid is emerging as a wide-scale infrastructure that promises to support resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organization [1].Grid computing applies the resources of many computers in a network for a single problem at the same time - usually to a scientific or technical one that requires a large number of computer processing cycles or access to large amounts of data. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing.

This chapter gives an overview of all aspects of grid computing. It discusses the technologies like distributed computing, parallel computing and various kinds of grid based services, different topologies of the grid and benefits of a grid environment.

## 1.1 Evolution of Grid Computing

Distributed computing is the combination of widely spread computational machines, to solve the large computative problems like, weather forecasting, satellite launching and earthquake predetermination etc. Distributed computing is another form of parallel computing where program parts, run on different machine simultaneously, in parallel computing program parts run simultaneously on multiple processors in the same computer. Both types of processing requires dividing a program into parts that can run simultaneously. Distributed programs often deal with heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers [3].

Distributed computing connects the two remotely situated machines via a program in which it makes call to other machine taking its address space as shown in Figure1.1. There are many nodes in distributed computing but a node in distributed environment does not know the hardware architecture on which the recipient is running, or the platform in which the recipient is implemented. There are many differences between local and distributed computing like throughput, memory access, concurrency and partial failure. In all these differences latency and memory access are well known and others are more difficult to explain [3].

The main problem in distributed computing is to manage all distributed resources by central resource manager along with managing issues related to performance, concurrency, communication, failure handling, deadlocks and security [3].

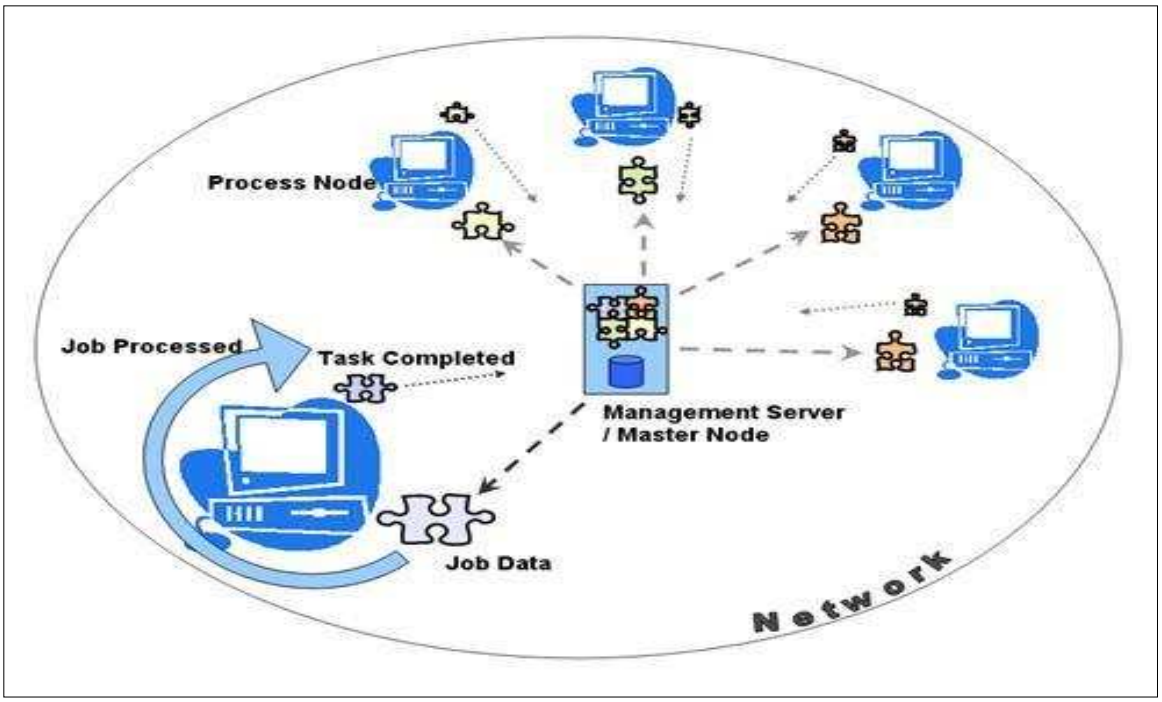


Figure 1.1 General View of Distributed Computing [57]

Moreover multiple point failure and more opportunities for unauthorized attack in distributed computing has paved way for grid computing which has thus evolved from distributed computing.

The Term “Grid” was coined in the mid 1990s to denote a proposed grid computing infrastructure for advanced science and engineering [15]. Back in 1965 the developers of an operating system called Multics (an ancestor of UNIX), presented a vision of "Computing as a Utility" - in many ways uncannily like the grid vision today. Access to the computing resources was envisioned to be exactly like access to utility such as electricity - something that the client connects to and pays for according to the amount of use. This led to coining of the term “Grid” [16].

The basic idea behind the grid is sharing computing power. Now-a-days most people have more than enough computing power on their own PC. A number of applications need more computing power that can be offered by a single resource or organization in

order to solve them within a feasible/reasonable span of time and cost. This promoted the exploration of logically coupling geographically distributed high-end computational resources and using them for solving large-scale problems. Such emerging infrastructure is called computational grid, and led to the popularization of a field called grid computing [15]. Table 1.1 below shows historical background of the grid.

Table 1.1 Historical Background of Grid [15]

| Technology | Year |
|---|---|
| Networked Operating Systems | 1979-81 |
| Distributed Operating Systems | 1988-91 |
| Heterogeneous computing | 1993-94 |
| Parallel and grid computing | 1995-96 |
| The grid | 1998 |

Cluster and Intranet computing are the parents of grid computing as shown in Figure 1.2. They came in existence, but there were many drawbacks due to which both technologies couldn't survive for a long time. There is collection of interconnected computers which are tightly coupled in case of cluster computing. This computing is cost effective but communication overhead, resource wastage and maintenance are its major drawbacks [54].

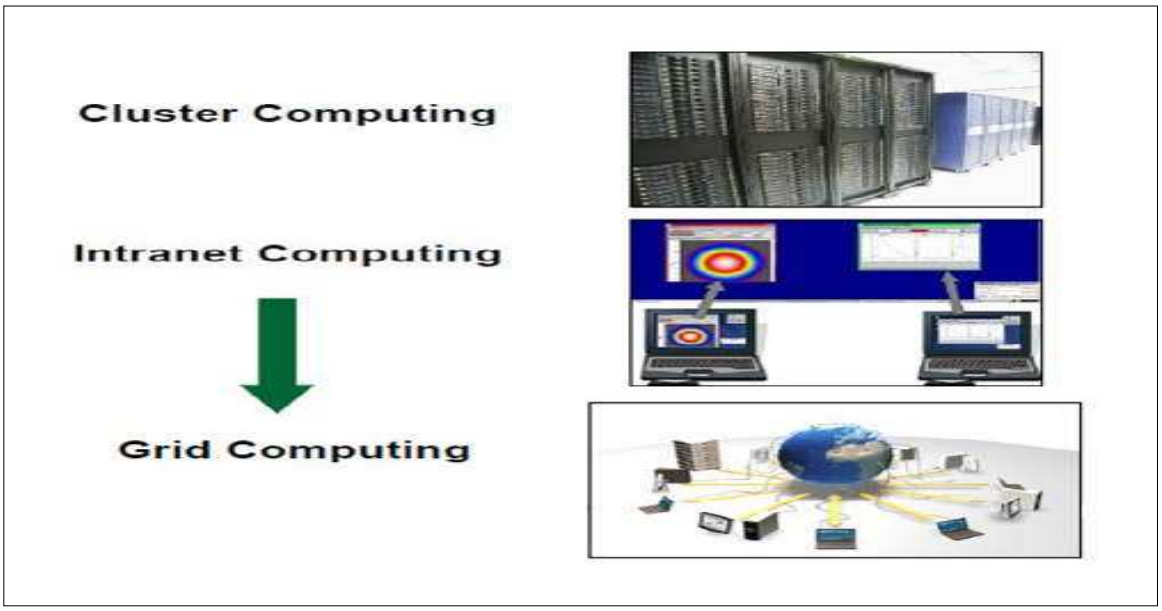


Figure 1.2 Evolution of Grid [54]

Intranet computing uses resources of a department network and management tool for running parallel jobs. In this computing, high reliability and scalability are due to high no. of resources used. Some drawbacks are lack of proper resource management and basic protocols and interfaces not based on open standards and no use of resources out of the domain of administration.

## 1.2 Characteristics of Grid Computing

In today's complex world computers have become extremely powerful and they are enough capable to run more complex problem, still there are many complex scientific experiments, advanced modeling scenarios, genome matching, astronomical research, a wide variety of simulations, complex scientific & business modeling scenarios and real-time personal portfolio management, which require huge amount of computational resources. To satisfy some of these aforementioned requirements, grid computing is being utilized [2].

Grid computing offers seamless access to distributed data and collaborative distributed environments, for running computationally intensive applications. Computing power available to any organization can thus be increased by using the idle periods of computing resources [1].

(a) Exploiting Underutilized Resources

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

(b) Parallel CPU Capacity

The potential for massive parallel CPU capacity [35] is one of the most attractive features of a grid. In addition to pure scientific needs, such computing power is driving a new evolution in industries such as the bio-medical field, financial modeling, oil exploration, motion picture animation, and many others. The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts.

(c) Collaboration of Virtual Resources

In the past, grid computing promised this collaboration and achieved it to some extent. Grid computing takes these capabilities to an even wider audience, while offering important standards that enable very heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of virtual resources. The users of the grid can be organized dynamically into a number of virtual organizations [1] each with different policy requirements. These virtual organizations can then share their resources collectively as a larger grid.

(d) Access to Additional Resources

In addition to CPU and storage resources, a grid can provide access to increased quantities of other resources and to special equipment, software, licenses, and other services. The additional resources can be provided in additional numbers and/or capacity. For example, if a user needs to increase his total bandwidth to the internet to implement a data mining search engine, the work can be split among grid machines that have independent connections to the internet [35].

(e) Reliability

Grid provides reliability in terms of failure at one location; the other parts of the grid are not likely to be affected. Grid management software can automatically or manually resubmit jobs most of the times 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, or tampering; thus offering much more reliability [35].

(f) Resource Balancing

A grid federates a large number of resources contributed by individual machines into a greater total virtual resource. 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 parts of a larger organization [1].

(g) Heterogeneity

A grid hosts both software and hardware resources that can be extremely diverse ranging from data, files, software components or programs to sensors, scientific instruments, display devices, personal digital organizers, computers, super-computers and networks [17]. A grid involves a variety of resources that are heterogeneous in nature and might span several administrative domains across wide geographical distances. Resources are owned and managed by different, potentially mutually suspicious organizations and individuals that likely have different security policies and practices [18].

(h) Scalability

It is a desirable property of a system, a network or a process, which indicates its ability to either handle growing amounts of work in a graceful manner, or to be readily enlarged. A grid might grow from few resources to millions [19]. This raises the problem of potential performance degradation as grid's size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant [20].

(i) Dynamicity or Adaptability

In a grid, a resource failure is the rule, not the exception. As in a grid there are numerous resources, the probability of some resource failing is naturally very high. The resource managers or applications must modify their behavior dynamically so as to extract the maximum performance from the available resources and services [20].

(j) Resource Coordination

Resources in a grid must be coordinated in order to provide aggregated computing capabilities [17].

(k) Reliable Access

A grid must assure the delivery of services under established Quality of Service (QoS) requirements. The need for reliable service is elementary since administrators require assurances that they will receive expected, continuous and often high levels of performance [20].

## 1.3 Types of Grid

There are different types of grid, providing computational capabilities, large data sets, infrastructure within the grid middleware and on-demand services. These are as follows:

(a) Computational Grid

Computational grid is hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities and that enables coordinated resource sharing within dynamic organizations consisting of individuals, institutions, and resources. Thus computational grid is designed so that users won't have to worry about where scientific and engineering computations are being performed [1].

(b) Data Grid

In present time grid application areas are shifting from scientific computing to industry and business applications, that's why data grids is an enhancement of computational grids, and have been designed to store,move,and manage large data sets exploited in distributed data-intensive applications [5]. Data Grid represents a combination of large data sets, currently terabytes and will grow to petabytes due to geographic distribution of users, resources and computational intensive analysis results in complex and strict performance demands that cannot be satisfied by ordinary data infrastructure [28].

(c) Semantic Grid

Semantic Grid is supported by two key building blocks – semantic web technologies for creating and maintaining models (ontology's), and semantic aware services and protocols for exchanging metadata [7]. Semantic Grid is an extension of grid and provides the infrastructure that systematically manages the complete lifecycle of metadata. Semantic Grid aims to provide an infrastructure within the grid middleware – providing a core set of services and protocols to support the sharing and management of all aspects of metadata. This will enable unanticipated reuse of grid services and resources, better support for interoperability, and flexible grids [6].

(d) Knowledge Grid

Knowledge Grid is an intelligent, sustainable internet application environment that enables people or virtual roles (mechanisms that facilitate interoperation among users, applications, and resources) to effectively captures, publish, share, and manage explicit knowledge resources [8]. It also provides on-demand services to support innovation, cooperative teamwork, problem solving, and decision-making.

(e) Service Grid

Service Grids are created in order to realize the business potential of web services. Service Grids represent distributed architectural component that realize an array of service business or utilities owning and deploying specific enabling services. Services businesses on the other hand, may either be specialized and independent businesses or revenue centers within larger enterprises offering their specialized enabling services to other enterprises [9].

## 1.4 Grid Protocol Architecture

Grid Protocol Architecture contains five layers which are shown in Figure 1.3 and described as under [11]:

(a) Fabric Layer

This layer is common low-level interfaces to computational and data resources, storage systems, catalogs, network resources and sensors e.g. resource descriptions, job control, file access, etc.

(b) Connectivity Layer

Main working of this layer is to provide communication between different layers and protocols. It exchanges messages and data streams using internet protocol stack.

(c) Resource Layer

Resource layer provide secure negotiation, initiation, monitoring, control, accounting and payment of shared operations on individual resources. It also handles of individual resources by using fabric layer function. It uses two classes of protocols, informational and management protocols.

(e) Collective Layer

This layer implements, different type of sharing behaviors on the resources, without placing new requirements. This helps to coordinate the multiple resources and works for general purpose to highly application and domain specific services also. Some collective services are directory services co-allocation, scheduling and brokering, monitoring, diagnostics grid-enabled programming systems workload management, community accounting and payment and software discovery.

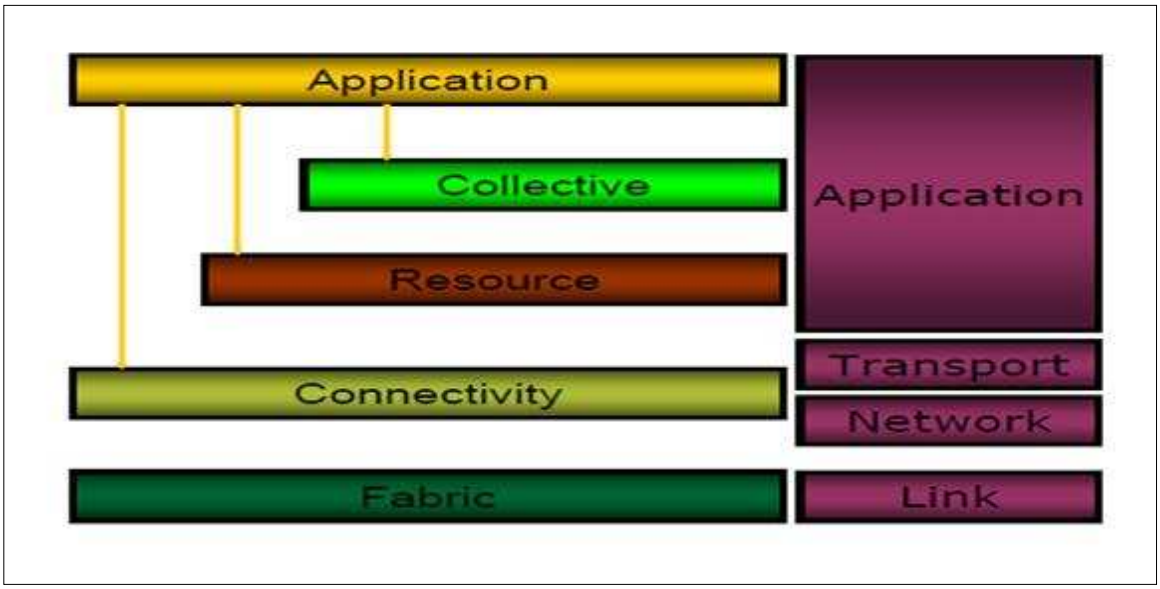


Figure 1.3 Grid Protocol Architecture vs. TCP/IP Protocol Architecture [54]

(d) User Layer

This layer provides easy user interface to user so that he could use the application easily. This uses the middleware API (application programming interface) and SDK (software development kit) for accelerated development.

## 1.5 Research Motivation

A typical distributed system will have a number of interconnected resources that can work independently or in cooperation with each other [13].Key characteristic of grids is to share resources (e.g. CPU cycles and network capacities)among numerous applications, therefore number of resources available to any given application highly fluctuates over time. In this scenario load balancing plays key role for those applications which are grid enabled. To minimize this unbalancing situation (time needed to perform all tasks), the workload should be evenly distributed over all resources based on their processing speed. Essential objective of load balancing consists primarily in optimizing the average response time of applications, which often means maintaining the workload proportionally equivalent on the whole resources of a system.

This work focuses on load balancing in a grid environment. Grid application performance is critical in grid computing environment. So to achieve high performance we need to understand the factors that can affect the performance of an application like load balancing, which is one of most important factors that influence the overall performance

of application. Although load balancing methods in conventional parallel and distributed systems have been intensively studied, they do not work in grid architectures because these two classes of environments are radically distinct and hence this area has been chosen in this research work [56].

## 1.6 Thesis Organization

Chapter 2 discusses all the existing categories, policies strategies of algorithms along with algorithms related to load balancing and job migration. Finally it discusses the problems found in existing load balancing and job migration algorithms.

Chapter 3 describes the problem formulation.

Chapter 4 discusses the factors for initiating load balancing algorithms, comparative study of existing load balancing algorithms, designing of proposed load balancing algorithms and implementations details.

Chapter 5 describes the experimental results of proposed algorithm.

Chapter 6 summarizes the work presented in this thesis followed by future scope of this work.

# Chapter 2 Literature survey

Grid functionally combines globally distributed computers and information systems for creating a universal source of computing power and information [13]. A grid can offer a resource balancing effect by scheduling grid jobs at machines with low utilization. A proper scheduling and efficient load balancing across the grid can lead to improve overall system performance and a lower turn-around time for individual jobs. Load balancing is required to disperse the resource's load evenly so that maximum resource utilization and minimum task execution time could be possible. This is very crucial concern in distributed environment, to fairly assign jobs to resources. This chapter provides the details about the existing load balancing algorithms and job migration in grid environment.

## 2.1 Load Balancing

Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to cut response time through an appropriate distribution of the applications [14]. To minimize the decision time is one of the objectives for load balancing which has yet not been achieved. Job migration is the only efficient way to guarantee that submitted jobs are completed reliably and efficiently in case of process failure, processor failure, node crash, network failure, system performance degradation, communication delay; addition of new machines dynamically even though a resource failure occurs which changes the distributed environment [23].

Generally, load balancing mechanisms can be broadly categorized as centralized or decentralized, dynamic or static, and periodic or non-periodic [21]. All load balancing methods are designed such as, to spread the load on resources equally and maximize their utilization while minimizing the total task execution time. Selecting the optimal set of jobs for transferring has a significant role on the efficiency of the load balancing method as well as grid resource utilization. This problem has been neglected by researchers in most of previous contributions on load balancing, either in distributed systems or in the grid environment [22].

There are different types of load balancing policies, strategies and categories which give different results to users in different environments or under different circumstances.

Some load balancing policies, category and strategies have been discussed in subsequent sections.

### 2.1.1 Load Balancing Categories

Load balancing problem has been discussed in traditional distributed systems literature for more than two decades and various algorithms, strategies and policies have been proposed, classified and implemented [28]. Load balancing algorithms can be classified into two categories, static and dynamic.

(a)Static Algorithms

Static load balancing algorithms allocate tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on average load of workstation cluster.

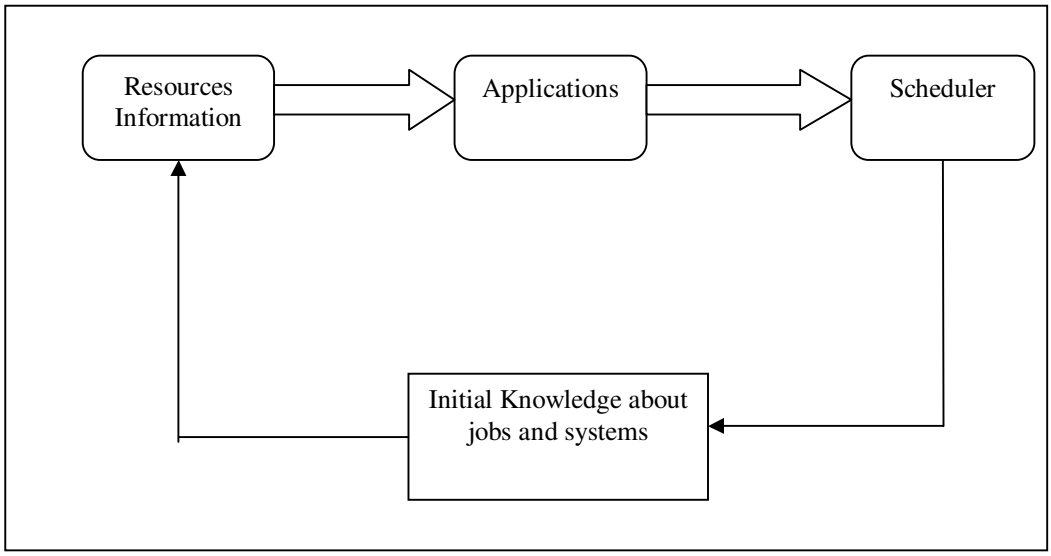


Figure 2.1 Static Load Balancing [28]

The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics [40].

However, static algorithms only work well, when there is not much variation in the load on the workstations. Clearly, static load balancing algorithms aren't well suited to a grid environment, where loads may vary significantly at various times. A few static load balancing techniques are [23]:

- Round-Robin Algorithm: tasks are passed to processes in a sequential order, when the last process has received a task the schedule continues with the first process (a new round) [59].
- Randomized Algorithm: allocation of tasks to processes is random [60].
- Simulated Annealing or Genetic Algorithms: mixture allocation procedure including optimization techniques [61].

Drawbacks of Static Load Balancing Algorithms

- It is very difficult to estimate a-priori (in an accurate way) the execution time of various parts of a program.
- Sometimes there are communication delays that vary in an uncontrollable way.
- For some problems the number of steps to reach a solution is not known in advance.

(b) Dynamic Algorithms

According to the name dynamic load balancing algorithms takes decision at run time, and use current or recent load information when making distribution decisions. In grid environment with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which determine when and which task has to be migrated [39].

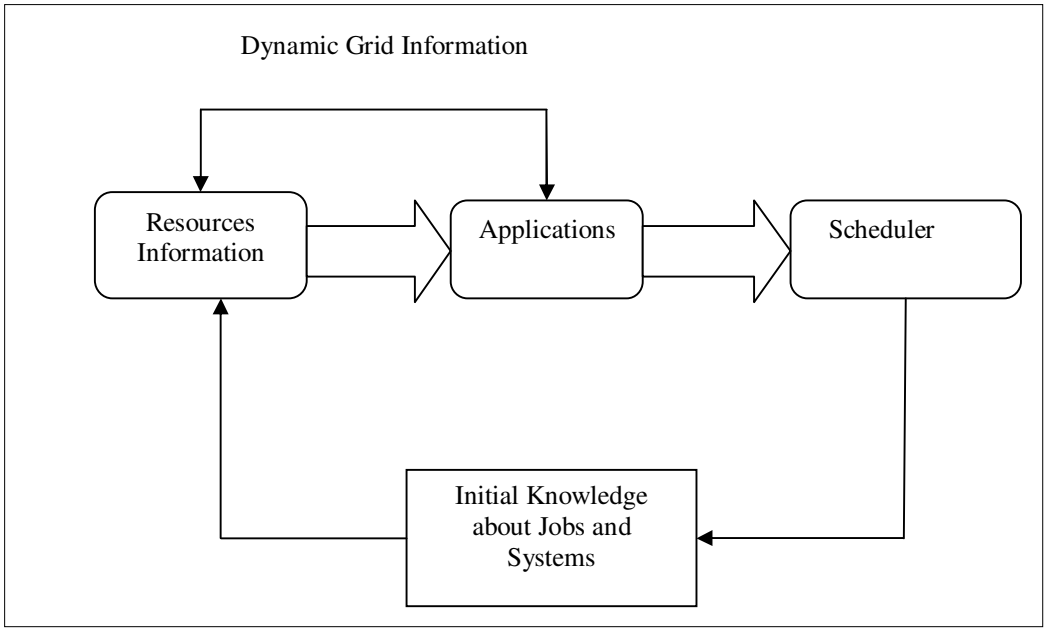


Figure 2.2 Dynamic Load Balancing [28]

After using effectively dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. But this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits [29].

### 2.1.2 Load Balancing Strategies

There are three major parameters which usually define the strategy of a specific load balancing algorithm [33]. These three parameters answer three major questions.

- Who makes the load balancing decision?
- What information is used to make the load balancing decision?
- Where the load balancing decision is made?

Some load balancing strategies are being discussed in the following section.

(a)Sender-Initiated v/s. Receiver-Initiated Strategies

In sender-initiated policies, congested nodes attempt to move work to lightly-loaded nodes. In receiver-initiated policies, lightly-loaded nodes look for heavily-loaded nodes from which work may be received [29].Figure-3.4 shows the relative performance of a sender-initiated and receiver-initiated load balancing algorithm. As can be seen, both the sender-initiated and receiver-initiated policies perform substantially better than a system which has no load sharing.

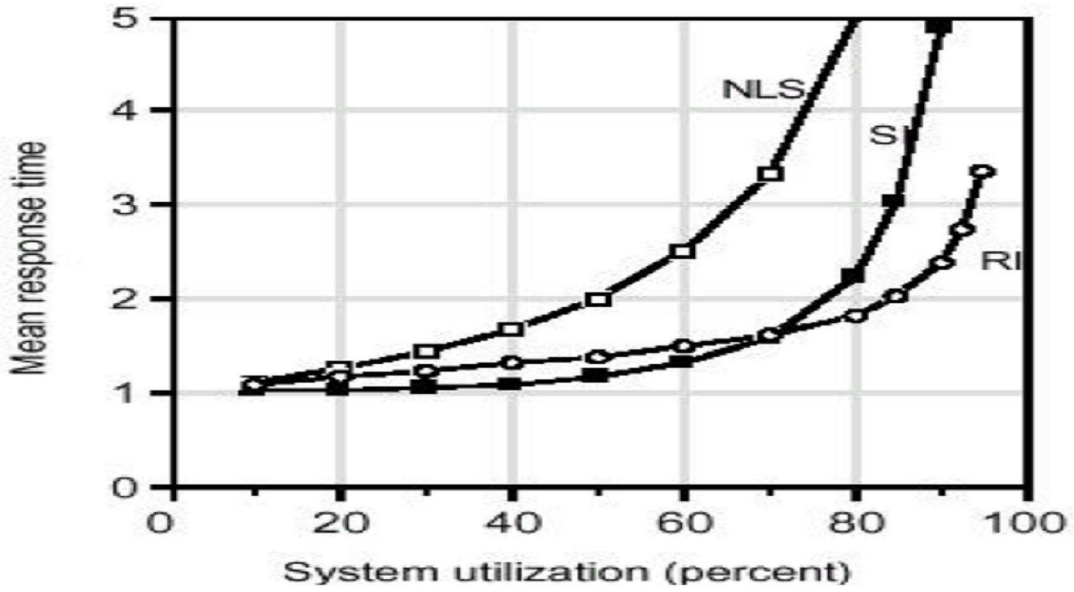


Figure 2.3 Performance of Sender-Initiated v/s Receiver-Initiated Strategies [29]

The sender-initiated policy performing better than the receiver-initiated policy at low to moderate system loads. Reasons are that at these loads, the probability of finding a lightly-loaded node is higher than that of finding a heavily-loaded node.

Similarly, at high system loads, the receiver initiated policy performs better since it is much easier to find a heavily-loaded node. As a result, adaptive policies have been proposed which behave like sender-initiated policies at low to moderate system loads, while at high system loads they behave like receiver-initiated policies.

(b) Global v/s. Local Strategies

Global or local policies answer the question of what information will be used to make a load balancing decision in global policies. The load balancer uses the performance profiles of all available workstations. In local policies, workstations are partitioned into different groups. The benefit in a local scheme is that performance profile information is only exchanged within the group. The choice of a global or local policy depends upon the behavior of an application, which will exhibit. For global schemes, balanced load convergence is faster compared to a local scheme since all workstations are considered at the same time [29].

Local and global scheme fall under the distributed scheme whereas centralized works under the global scheme. In local strategy every node uses the local information of neighbor node to balance the node load. Main motive of this strategy is to reduce the remote communication overhead without using much information as global strategy. In global strategy more and more information is required to take appropriate decision according to load distributed there, that's why scalability is low in global case [34].

However, this requires additional communication and synchronization between the various workstations; local schemes minimize this extra overhead but reduced synchronization between workstations is also a downfall of the local schemes, if the various groups exhibit major differences in performance. If one group has processors with poor performance (high load), and another group has very fast processors (little or no load), the later will finish quite early while the former group is overloaded.

(c) Centralized v/s. De-centralized Strategies

A load balancing strategy is categorized as either centralized or distributed, both these define where load balancing decisions are made [30]. In a centralized scheme, algorithm is located on one master workstation node and all decisions are made there.

Basic features of centralized approach are:

- There is master node which holds collection of tasks and knows what have to do with particular task
- Select the node according to task
- After execution it takes another task.

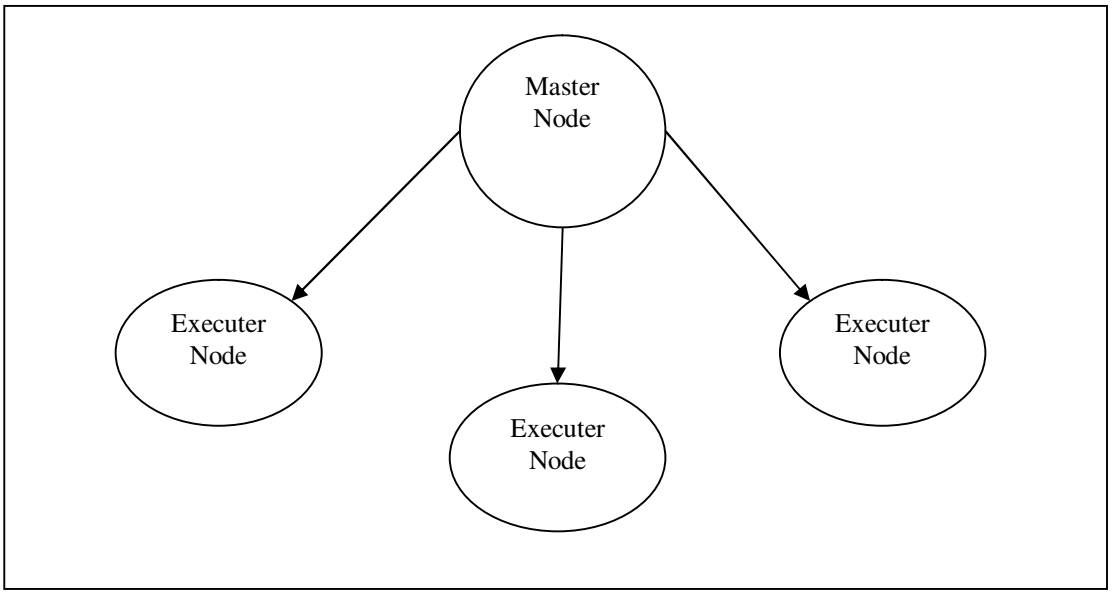


Figure 2.4 Centralized Strategies [30]

For centralized schemes, the reliance on one central point of balancing control could limit future scalability. Additionally, the central scheme also requires an "all-to-one" exchange of profile information from workstations to the balancer as well as a "one-to-all" exchange of distribution instructions from the balancer to the workstations. The distributed scheme helps, to solve the scalability problems, but at the expense of an "all-to-all" broadcast of profile information between workstations. However, the distributed scheme avoids the "one-to-all" distribution exchange since the distribution decisions are made on each workstation [31].

In a de-centralized scheme, the load balancer is replicated on all workstations. There are different algorithms used in de-centralized scheme for job selection. These algorithms are round- robin algorithm, random polling algorithm etc [32].

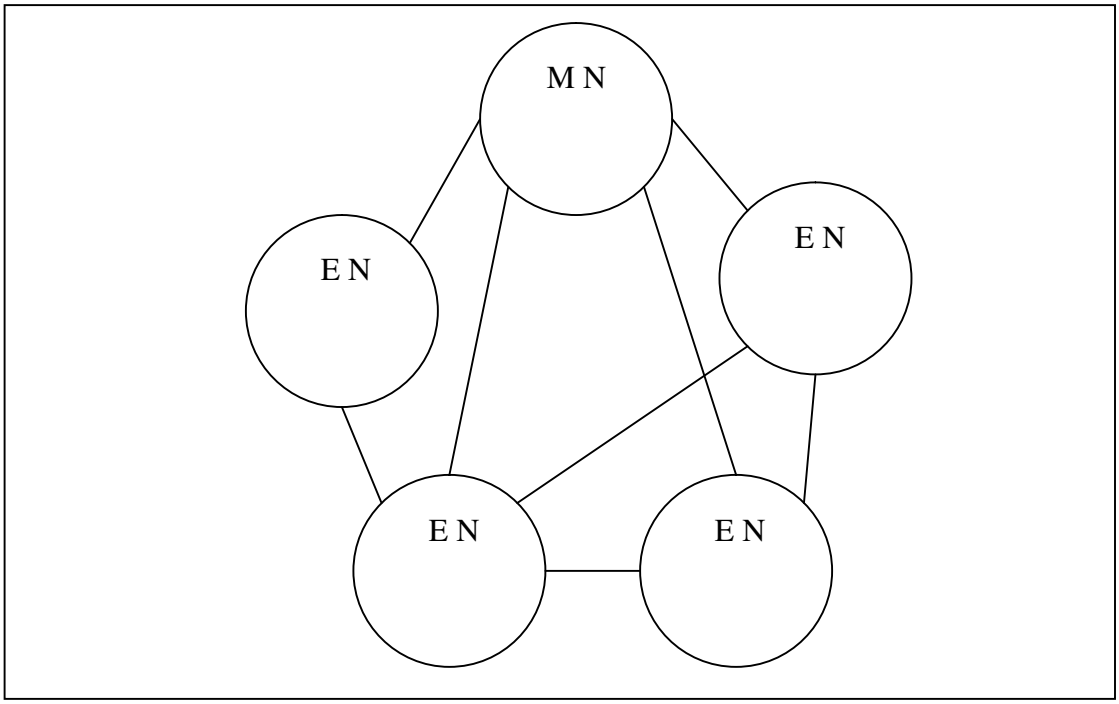


Figure 2.5 Decentralized Strategies (E N executer node, M N master node) [32]

(d) Co-operative v/s Non co-operative

In case of co-operative strategy node at which load is being balanced by other node, should co-operate the other node. If it doesn't do that it would be non co-operative strategy. In this it takes its own decision to balance the load [34].

(e) Adaptive vs. Non-adaptive

In adaptive system, it considers the predetermined decision, past and current system information affected by the previous decision and changed environment and parameters. In non adaptive environment parameters used in scheme remain same regardless to system past behavior [34].

(f) One time assignment vs. Dynamic Reassignment

In one time reassignment job are assigned to resource only one time and not migrated to other node whether it is completed or not. But in dynamic reassignment job are migrated

among the different node until it uncompleted. Disadvantages of dynamic reassignment are that job may be incomplete in such migrating process [34].

### 2.1.3 Load Balancing Policies

Load balancing algorithms can be based on many policies; some important policies are defined below [30].

- Information policy**:** This policy specifies what workload information should be collected, when it is to be collected and from where.
- Triggering policy**:** This policy determines the appropriate period to start a load balancing operation.
- Resource type policy**:** This policy classifies a resource as server or receiver of tasks according to its availability status.
- Location policy**:** This policy uses the results of the resource type policy to find a suitable partner for a server or receiver.
- Selection policy**:** This policy defines the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).

The main objective of load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way. Hence it is significant to define metrics to measure the resource workload. Every dynamic load balancing method must estimate the timely workload information of each resource [32]. This is the key information in a load balancing system where responses are given to following questions

- How to measure resource workload?
- What criteria are retaining to define this workload?
- How to avoid the negative effects of resources dynamicity on the workload?
- How to take into account the resources heterogeneity in order to obtain an instantaneous average workload representative of the system?

Success of a load balancing algorithm depends upon stability of the number of messages (small overhead), support environment, low cost update of the workload, and short mean response time which is a significant measurement for a user. It is also essential to measure the communication cost induced by a load balancing operation, but to achieve all these, anyone would have to face great challenge in grid environment [31].

## 2.2 Challenges of Load Balancing in Grid Computing

There are many challenges related to load balancing in grid environment. Some are discussed further:

(a) Resource Heterogeneity

There are two types of resources in computational grid, first one are network resources and second one are computational resources in which heterogeneity exists. In networks resources it may be in terms of bandwidth and used network protocols. In case of computational resource there may be different hardware, architecture, no. of resources, physical memory size, CPU speed and so on. Heterogeneity results in differing capability of processing, which is the main cause for load balancing in heterogeneous environment [1].

(b) Site Autonomy

Grid with their distributed ownership and cross-domain organization gives a different type of problem in grid environment. As a direct result of distributed ownership, resource owners take management decision regarding local resources, often based on local information and local requirements. If the whole system would be shared there, then security of the system may leak. An application from the unauthorized user can't be run on any system. To efficiently manage the resources available in grid systems to meet the needs of an ever-changing and diverse user community, an automated, agile, and adaptive dynamic control system with grid-wide perspective is needed [3].

(c) Dynamic Behavior

In grid environment, there are more and more problems due to the heterogeneity of resources. Any time any resource may be available and can be unavailable due to machine failure or connection problems. This dynamic behavior always gives headache to the user of grid environment [3].

(d) Resource Non-Dedication

Due to non dedication of resource, resource may join the many grid systems simultaneously. A contention arises there when requests from many users come there. So due to resource non dedication resource usage contention is the major issue in grid environment.

(e) Application Diversity

In grid environment there is different types of users having different type of applications and has different requirements. For example some applications may have set of dependent jobs, other may have independent jobs, and on other side some application requires sequential execution. Keeping all aspects of jobs in mind, designing a general purpose load balancing system is extremely difficult [34].

(f) Resource Selection and Computation Data Separation

Resource selection in grid environment is difficult due to large no of resources, which are dynamically being joining grid and some are leaving the grid due to machine failure or connection problem. That's why the cost data staging is high in grid environment, compare to traditional systems. Data staging cost is most prominent obstacle in grid environment or a main problem for designing the efficient and most effective load balancing system in grid environment [34].

## 2.3 Job Migration

Job migration means re-allocation of jobs from one system to another system or in other words Job migration is the process of moving a normal, checkpointable or rerunable job from one host to another [62]. This facilitates load balancing by moving jobs from a heavily-loaded host to a lightly-loaded host. Job migration can be initiated manually or automatically. To initiate job migration automatically, configuration of a migration threshold at job submission, or at the host, queue, or in an application profile is required. Job migration is a solution of the load misbalancing problem because it is required to balance the load whenever load imbalance problems come in distributed environment. There are many algorithms related to the load balancing and job migration which is discussed further [61]. Figure 2.6 show how job migration occurs.

## 2.4 Job Migration and Load Balancing Algorithms

Grid is such a system, where environmental conditions are subject to unpredictable changes: system or network failures, system performance degradation, addition or deletion of new machines, variance cost of resources, etc. In such conditions, job migration is the only efficient way to guarantee that the submitted jobs are completed as per user requirements [42].

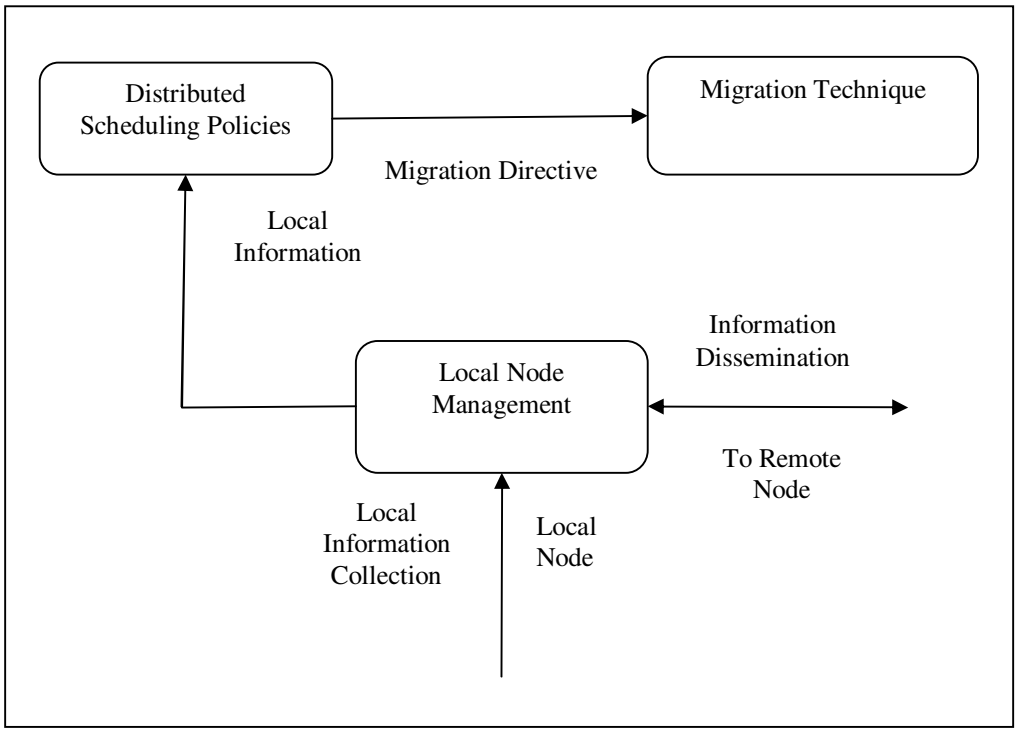


Figure 2.6 Job Migrations [62]

The major tasks involved in migration are job monitoring, re-scheduling, and check-pointing. Job monitoring is responsible for detecting alert situations and information about the systems that would be helpful to initiate a migration. This information is reported to the re-scheduler, which evaluates if it is worth migrating the job, and in that case, decides a new allocation for the job. Check-pointing is the capability of capturing periodically a snapshot of the state of a running job, in such a way that the job can be restarted from that state in a later time in case of migration. Many systems which are dealing with job migration generally face up some problems from the point of view of performance [41].The main migration policies which can be considered are performance slowdown, target system failure, job cancellation, detection of a better resource, etc. New job migration policies can be based upon, like the discovery of a new cheaper resource, or variations in the resource prices during the job execution [43].

### 2.4.1 Migration Mechanism

There are many migration mechanisms, like shared-storage device, preserving memory image, initial placement, preemptive migration, user level migration, kernel level migration, homogeneous and heterogeneous migration are discussed in this section.

(a)Shared-Storage devices v/s Preserving Memory Image

This is the most common method used in a migration mechanism. Once the system detects load imbalance of some processors over specific threshold, it migrates the tasks or jobs in the waiting queue, to the most suitable idle processor through shared-storage devices, which acts as a media for the storage of process states or images [40].

Preserving memory image means save and restore which comes under check-pointing. Check-pointing is combination of two activities saving the running data and restoring it after getting suitable resource. Captured states and data of a running process include, registers containing the address, variables, and data else, memory spaces keeping source codes, libraries, data structures, files containing data with a large size. There are two types of check-pointing kernel-level and user-level. Kernel-level check-pointing are a part of the operating system kernel and specific for some kernel version while user-level check-pointing is application programs and implemented through interprocess communication mechanisms such as signals, making user-level check-pointing portable[46].

(b) Initial Placement vs. Preemptive Process Migration

Remote execution or Initial placement is a very simple mechanism to get system-wide utilization of available resources within a distributed environment. This mechanism creates the particular process on a remote machine prior to execution. Sometimes this involves the transfer of code or process environments, such as opened files. Usually remote execution is faster than process migration because it does not transfer potentially large amounts of process information [45].

Major disadvantage of remote execution is the lack of flexibility and transparency to the process, which can only be moved at the time of its creation. Otherwise additional information about the behavior of the process can be obtained if the process runs for a period of time on the source machine. This additional information can be used to make more appropriate load balancing decisions.

Preemptive process migration dynamically reallocates a running process to another machine in a distributed system at an arbitrary time after initiating execution at the source machine. Information that has to be transferred depends on the employed migration algorithm. Larger information compared to remote execution is transferred here because

information can consist of the entire process environment including the process address space. One advantage of this mechanism is that after the process begins, changes in load of a system can be estimated so that load balancing policies can take more efficient decisions that's why preemptive process migration leads to a better system-wide utilization of available resources compared to remote execution [47].

(c) User-Level vs. Kernel-Level Process Migration

Process migration can be applied to different levels of an operating system where it results in varying performance levels, fault resilience, and re-usability. Existing implementations of process migration are done at kernel-level or at user-level, including implementations as a part of an application.

Implementation of user-level process migration is simpler than kernel level because there is no need of change in the underlying operating system. An example of a user-level process-migration implementation is Condor [48]. Kernel-level process migration involves modifications and extensions to the underlying operating system kernel which leads to additional complexity. Therefore, deployment is more difficult than with a user-level implementation. On the other hand, the direct and fast access to kernel information about the migrating process results in smoother performance. Another advantage of kernel-level process migration is transparency to the client application, which does not need to be changed or designed with process migration in mind [47].

(d) Homogeneous v/s. Heterogeneous Process Migration

Homogeneous process migration stands for migrating processes in a homogeneous environment of a distributed system. A process can be migrated only among machines with the same compatible architecture and operating system but does not necessarily have the same resources and capabilities. Most systems providing process migration are restricted to homogeneous process migration, including MOSIX [49], RHODOS [50] and Sprite [51].

A single computer network often consists of machines that may vary in their architecture and operating systems. With heterogeneous process migration it is possible to migrate a process among such dissimilar machines. Obviously, the implementation of process migration in a heterogeneous environment leads to significantly more complexity and introduces performance penalties due to costly translations. Architecture and operating-

system-specific features must be considered here. Additionally, the state of a process must be represented in a machine-independent way to be transferred and resumed. Systems implementing heterogeneous process migration are Emerald [52] and TUI [53].

### 2.4.2 Load Balancing Mechanism

There are some load balancing algorithms like virtual machine migration, node reconfiguration by user level thread migration, robin-hood an active objects migration mechanism for intranet, load based graph method and data consolidation. All there are discussed further in details.

(a)Virtual Machine Migration (Live Migration)

In virtual machine migration snapshots of machine are sent to other machine that's why it is called the virtual machine migration. There are two methods for virtual machine migration. First one is live migration and second one is regular migration [24]. In live migration, running domain between the different host machines is migrated without stopping the job. In between it stops job and gathers all required data then resumes. But this happens only in same layer-2 network and IP subnet. In regular migration generally stop the job then migrated.

An important aspect of this mechanism is to make the run-time job migration with non-dedicated shared resources in dynamic grid environment. Virtual machine migration provides high isolation, security and customization environment in which administrator privileges the user to execute the work. EtherIP and IP tunneling are required while migrating in this mechanism. This algorithm redistributes the load coming to any particular node, which may be the old connected node or newly added node for that load.

(b) Node Reconfiguration by User Level Thread Migration

This mechanism makes application workload migrate from source node to destination node, and then let source node depart from original computing environment .There are two mechanism for this, first one is node reconfiguration by user-level thread migration and another one is node reconfiguration by kernel level thread migration. Node reconfiguration by user level thread migration has been discussed in this survey.

There are two implementation methods of node reconfiguration. One is synchronous method and the other is asynchronous method. In synchronous method, all nodes are paused during reconfiguration. On the other hand, in asynchronous method all nodes

continue to work simultaneously with reconfiguration. Synchronous method may make performance down even though it is easier to design. Alternatively, better performance can be obtained by asynchronous method as long as more attention paid to correctly maintain the order of node reconfiguration messages [24].

Information regarding redistribution of workload and how to add/delete nodes is present in the implementation of node reconfiguration mechanism. With the help of user level thread migration, which is already supported by the thread package workload, is redistributed here. Same as virtual machine migration, node reconfiguration mechanism also needs to transfer in memory states from source node to destination node.

(c) Check-Pointing Approach

Checkpoint is defined as a designated place in a program at which normal processing is interrupted specifically to preserve the status information necessary to allow resumption of processing at a later time. By periodically invoking the check pointing process, one can save the status of a program at regular intervals. If there is a failure one may restart computation from the last checkpoint thereby avoiding repeating the computation from the beginning. The process of resuming computation by rolling back to a saved state is called rollback recovery [25].

There are three types of check pointing implementations, kernel-level, user-level and application-level. These implementations differ in level of transparency, efficiency and mechanism used to initiate checkpoint and restart. In kernel level check pointing user does not have to change the application at all so least efficient, because system does not have the knowledge about the application. Developer achieves user level check pointing, and he puts or implements some set of procedures that handle check pointing and restart. Developer knows all about the application that's why this approach is more efficient. The developer itself achieves application-level check pointing. This approach is the most efficient, because developer has detailed knowledge about application.

This is very useful in case of preemption and migration and is used in making fault tolerant systems. Most common benefits of the check pointing technology are the high level of fault tolerance offered by the applications that can be check pointed. Besides it used to recover from failures, playback debugging distributed programs, migrating processes in a multiprocessor system, software rejuvenation and optimistic simulation.

Check- pointing balances the load of processors in a distributed system; processes are moved from heavily loaded processors to lightly loaded ones. Check pointing process periodically provides the information necessary to move it from one processor to another.

(d) Robin Hood: An Active Objects Load Balancing Mechanism for Intranet

Robin-hood algorithms present a new totally non-centralized solution, multicast channel to communicate, and synchronize the processors and proactive tools to migrate jobs between them. Proactive techniques are very useful and provide the mobility and security in uniform framework. This work focuses on dynamic load balancing. Main objective of this algorithm is to improve the decision time in non-centralized environment. In this mechanism two basic things have been considered, first one to know about the local load and second one to transfer the load from high dense node to the less loaded node. This uses the non-centralized architecture and non-broadcasting of the balance of each node to reduce the overload in network. This is totally non-centralized load balancing mechanism, using the proactive library for the migration of jobs, and a multicast channel for node coordination [14].

(e) Load Graph Based Transfer Method

Load based graph method is based on network graph where each node is represented with its load, whereas load can be the number of users, average queue length or the memory utilization. It uses analytic model and single load determination policy throughout the system and load is determined on the basis of memory utilization and average queue length. This algorithm is based on three-layered structure. Top layer is load balancing layer which takes care of token generation, taking decision about task transfer, middle one is called monitoring layer and acts as an interface between top and middle and monitors load changes and third one called communication layer which take care of actual task transfer.

Here token is generated on the basis of outgoing and incoming edges and initialized on the basis of some specific value HWM & LWM (Highest Water Mark, Lowest Water Mark). Specific values are decided on the basis of load value of neighbors. Nodes having load value greater than HWM and are local maxima or nodes having load value less than LWM and are the local minima, can initiate token [26].

Maximum message transfer per node, if N is number of nodes and X is maximum message transfer per node Total message transfer =NX

And transfer of task will occur only if there would be proper load difference between the nodes would be as La - Lb > M where M is the required load difference for the task transfer.

Token will be generated if following conditions will be satisfied

(1.) For nth node (Load) n > L where L is maximum Load where load balancing is not required.

(2.) (Load) n > ∑ sum of load of all nodes

If both conditions are satisfied then the token is generated in more than sixty percent of the cases where load imbalance exists token finds out the proper node for the task transfer which improves the system performance. In this algorithm along with the task transfer among the neighboring nodes with the token transfer method care is taken to avoid the starvation of those nodes for which neighbors are not suitable for the task transfer [26]. The major parameter, network-partitioning issues along with inter-cluster and intra-cluster transfers for decision-making of load balancing for the transfer is considered here. There are some job migration algorithms and strategies in this section which will show how these algorithms works and which procedure these follow.

(f) Data Consolidation: A Task Scheduling and Data Migration Technique for Grid Networks

Generally grid contains all the heterogeneous and homogeneous geographically distributed computing and storage resources which may belong to different user or domain but are shared among users by establishing global management. When anybody uses the data on large scale within the grid that is called the data grid like in life sciences, high-energy physics and astrophysics, where large amounts of data are created, processed and stored in a distributed manner [27].

But in above applications network communication delays occur and it considerably increase task completion time that's why collaboration of task scheduling and data management is essential for increasing performance in such an environment [27].In this algorithm data replication technique is considered here which provides fast access and reliability by reducing the scheduling of task and data management. This is the main

optimization technique; provide fast data access and reliability. In this, data are scattered throughout the Grid network so that anyone could get it easily and efficiently.

Data consolidation (DC) is the combine name of task scheduling and data migration which have been taken here. This applies only to the data intensive applications that need many replicas of data for their execution and if DC is performed efficiently, important benefits can be obtained in terms of task delay, network load and other performance parameters of interest. DC can be considered as the dual of the data replication problem, where from one site the data are scattered to many repository sites. Through this dual relationship we will show that DC can also provide, by reversing the procedure, data replication services.

## 2.5 Conclusion

This chapter discusses all the existing load balancing algorithms, working under different strategies, policies and categories. Both static and dynamic algorithms give different performance results in customized environment. Static algorithms always give the constant performance but dynamic algorithms are preferred over static due to heterogeneous and dynamic environment. Next chapter, chapter 3 discusses the problem formulation.

# Chapter 3 Problem Formulation

Grid computing allows selection, sharing and aggregation of large collections of geographically and organizationally distributed heterogeneous resources for solving large-scale data and compute intensive problems. Load of resources varies with change in configuration of grid and it makes load balancing one of the major challenges in case of grid environment. This chapter presents detailed description of thesis problem and analysis of the gaps in existing research works.

## Problem Statement

In grid environments, the shared resources are dynamic in nature, which in turn affects application performance. Workload and resource management are two essential functions provided at the service level of the grid software infrastructure. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. Migration of jobs is one of the best ways to balance the load. The focus of this study is to consider factors which can be used as characteristics for decision making to initiate load balancing as it is one of the most important factors which can affect the performance of the grid application.

The main objectives of this thesis are:

1. To analyze the factors due to which Load balancing is required in grid environment.
2. A comparative study of existing load balancing algorithms.
3. To propose and design a load balancing algorithm for grid environment.
4. To present job migration technique for balancing load.

Next chapter presents the factors for initiating load balancing algorithms, comparative study of existing load balancing algorithms and proposed load balancing algorithm.

# Chapter 4 Proposed Load Balancing Algorithm and Job Migration Technique

Load balancing is defined as the feasible allocation or distribution of the work to highly feasible resources wherever it suits and execution time of the application could be minimized. This chapter discusses the factors for initiating load balancing algorithms, comparative study of existing load balancing algorithms and design of proposed load balancing algorithm.

## 4.1 Factors for Initiating Load Balancing Algorithm

There are some decision making factors, like arrival and completion of job, arrival and withdrawing of resources, on occurrence of which the load balancing algorithm initiates. Other factors are machine failure and node overloading which have surfaced from literature survey. These factors can be summarized as:

(a) Arrival of any new job: Whenever any new job arrives to scheduler, it demands for a resource which initiates load balancing algorithms.

(b) Completion of execution of any job: Whenever any new job gets complete, it leaves resources and they are reassigned to another job.

(c) Arrival of any new resource: Whenever any idle resource comes to systems resource pool then it is assigned to some job considering its requirement.

(d) Withdrawal of any existing resource: Withdrawing of any resource increases the computation overhead over remaining resources.

(e) Machine failure at any node: Failure of any machine increases the overhead of remaining resources that initiate load balancing algorithms.

(f) Node become overloaded: Whenever any node becomes overloaded due to any reason then load balancing algorithms starts and manages the stopped job of the overloaded node.

## 4.2 Comparative Analysis of Existing Load Balancing Algorithms

In this section comparative study of load balancing algorithms (already mentioned in literature survey) has been done on the basis of some parameters like, scalability,

security, throughput, migration time, efficiency, fault tolerance, overload rejection, resource utilization, forecasting accuracy and stability etc.

- Node reconfiguration mechanism makes application workload migrate from source node to destination node, and then let source node depart from original computing environment. There are two implementation fashions of node reconfiguration, first one is synchronous method and the other is asynchronous method. Best performance can be obtained by maintaining the order of node reconfiguration messages [24].
- In Virtual machine migration snapshots of machine are sent to other machine by using two migration method, live migration and regular migration. An important aspect of this mechanism is to make the run-time job migration with non-dedicated shared resources in dynamic grid environment and also provides high isolation, and security [24].
- Robin Hood algorithm is totally non-centralized mechanism, using the proactive library for the migration of jobs, and a multicast channel for node coordination. It improves the decision time in non-centralized environment and uses the non-centralized architecture and non-broadcasting of the balance of each node to reduce the overload in network [14].
- Load based graph method is based on network graph where each node is represented with its load, whereas load can be the number of users, average queue length or the memory utilization. It uses analytic model and single load determination policy throughout the system and load is determined on the basis of memory utilization and average queue length. The major parameter, network-partitioning issues along with inter-cluster and intra-cluster transfers for decision-making of load balancing for the transfer is considered here[26].
- In Data Consolidation algorithms data replication technique is considered here which provide fast access and reliability by reducing the scheduling of task and data management. This is the main optimization technique; provide fast data access and reliability. Data consolidation (DC) is the combine name of task scheduling and data migration. DC can be considered as the dual of the data replication problem,

where from one site the data are scattered to many repository sites. Through this dual relationship we will show that DC can also provide, by reversing the procedure, data replication services [27]. Table-4.1 shows the comparative study of all the existing algorithms described above:

Table 4.1 Comparative study of existing algorithms

| Algorithms / Parameters | Node Reconfiguration by User Level Thread Migration | Virtual Machine Migration(live migration | Robin Hood: An Active Objects Load Balancing Mechanism | Load Graph Based Transfer Method | Data Consolidation |
|---|---|---|---|---|---|
| **Efficiency** | Medium | Good | Not good | Good | Medium |
| **Overload Rejection** | No | No | No | Yes | No |
| **Stability** | Less | Less | Large | Large | Large |
| **Scalability** | High | High | Not good | Medium | High |
| **Throughput** | Low | High for small jobs | medium | low | Medium |
| **Resource Utilization** | Less | Less | Less | High | High |
| **Forecasting Accuracy** | Less | Medium | More | More | More |
| **Fault Tolerance** | Medium | High | Less | Medium | High |
| **Security** | Not good | High security | High security | No security | No security |
| **Performance** | High performance in intra cluster scenario | High performance in inter cluster scenario | Based on percentage loading at node | High | High due to data replication |

The above comparative study shows that all the characteristics all that fulfilled by any single load balancing algorithm as this is very difficult to achieve. Parameters like performance, throughput, resource utilization and efficiency are mandatory and therefore should be considered always while developing dynamic load balancing algorithms. In the

proposed algorithm discussed in next section, the parameters have been considered to a large extent for a grid environment.

## 4.3 Proposed Load Balancing Algorithm

Proposed load balancing algorithm is developed considering main characteristics like performance, throughput, and resource utilization.

### 4.3.1 Architecture of Load Balancing System

This section discusses about the architecture of load balancing algorithm-imposed system. Figure 4.2 presents a pictorial view of the system. Monitor server uses monitoring tool to gather information about all the connected nodes. This resource information is managed, processed and updated to a database. This information is accessed through web pages and is presented to the users. The web pages can be accessed from any nodes at the same network.

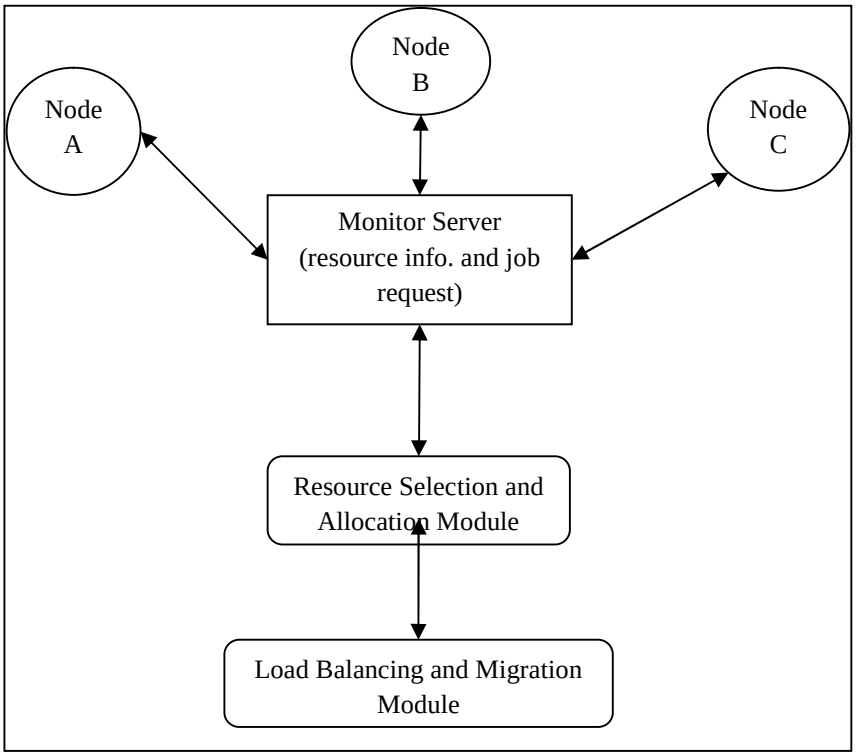


Figure 4.1 System Overall Architecture

Server node gets all information from the nodes via monitoring tool and updates the database. Web pages are created and hosted on local network using Apache HTTP server to provide the resources information to users, any system on local network running

Apache HTTP server can access these web pages. This is shown in figure:

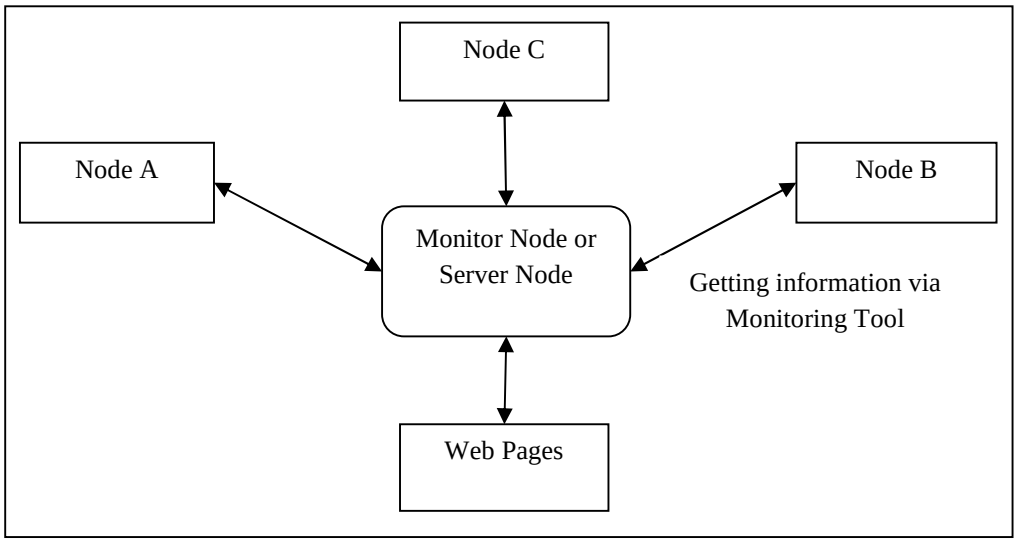


Figure 4.2 Implementation View of Algorithm

### 4.3.2 Design of Proposed Load Balancing Algorithm

This section discusses pseudocode and flowchart of the proposed load balancing algorithm. Proposed algorithm for load balancing is given below:

```
Begin
(Initiating activity happens)
(Load balancing start)
Monitoring info. and job request (n resources & n jobs)
Create job queue
If (job req. not matched with resource.)
        Job goes to main queue
Else
        Assigned job to resource
If (Machine failure)
         Job goes to main queue with check-pointed data
Else
        Job completed
End
```

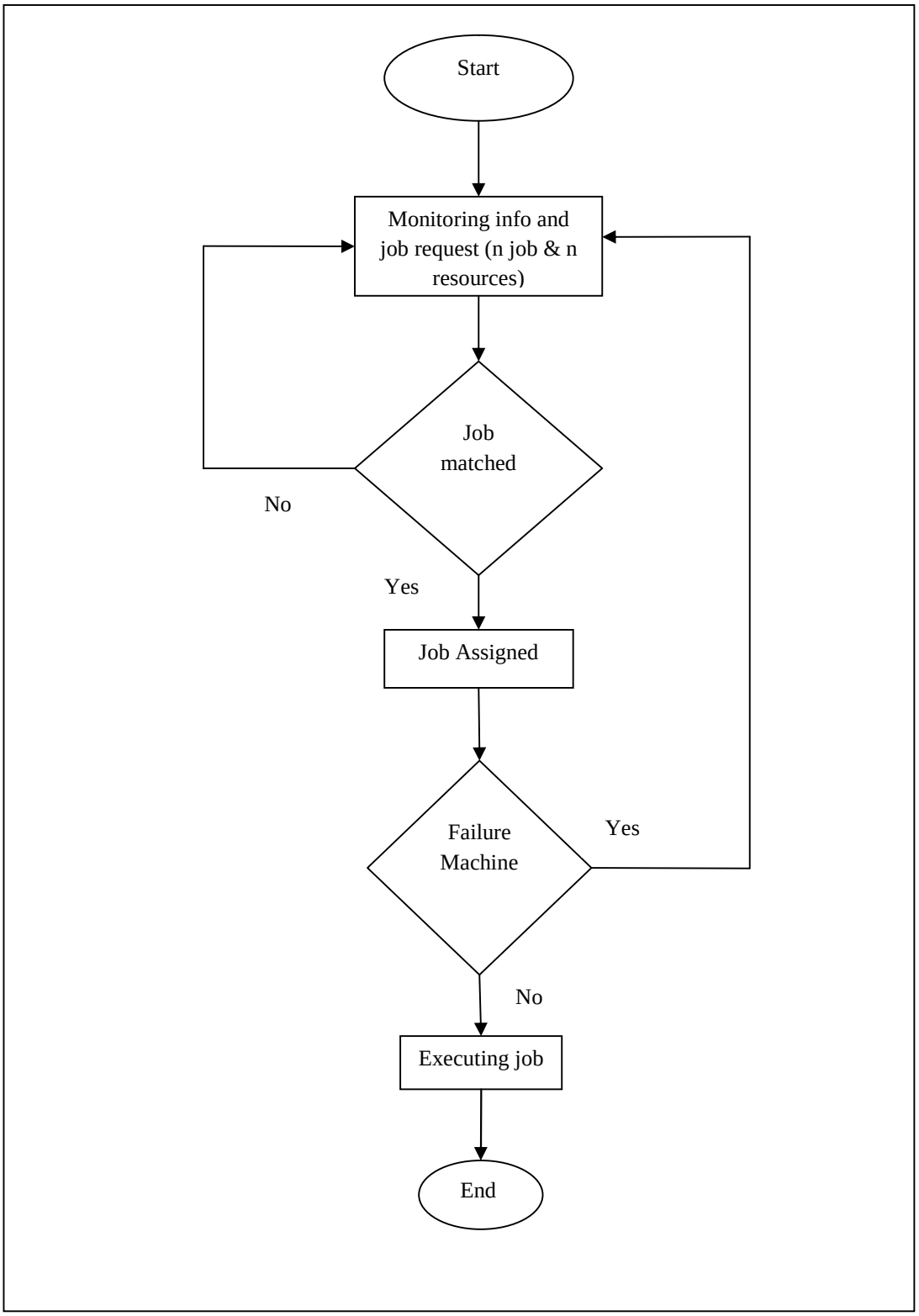


Figure 4.3 Flowchart of Proposed Algorithm

### 4.3.3 Complexity of Proposed Load Balancing Algorithm

Complexity is a measure of the performance of an algorithm in term of CPU time and memory usage. In this case computational complexity has been considered as this algorithm is for grid environment.

Computational Complexity: To measure computational complexity computation number should be known.

By Big-O notation

Above proposed algorithm have equation like this

$N^3+N^2+N+C$, where C is constant

According to Big-O notation

$f(n) = O(N^3) + O(N^3) + O(N) + O(C)$ ,C is constant

$f(n) = O(N^3)$

Another formula to find out complexity of algorithm

Complexity = No. Of closed loop = 3;

Another formula to find out Complexity of algorithm

Complexity = No. of decision making statements + 1;

Complexity = 2 +1 =3;

## 4.4 Implementation Details

This section discusses about the implementation details of the load balancing algorithm based system. Fedora Core 8 is used as operating system in implementation.

### 4.4.1 Technologies Used

To implement the proposed load balancing algorithm, an application has been developed, which is executed in simulated grid environment. To develop application J2EE and MySQL database server is used to maintain the monitoring data. The website is designed using PHP and hosted on local network using apache HTTP server.

(a) Globus Toolkit

Globus Toolkit 4.2.1 is used to create grid for deploying and testing the proposed monitoring system. Globus Toolkit is developed and provided by Globus Alliance. Globus Toolkit includes software and libraries for resource monitoring, resource management, data management, communication, fault detection, security and portability.

GT4 uses Web services mechanism heavily to provide flexible, extensible, and widely adopted XML-based mechanisms for describing, discovering, and invoking network services. Globus Toolkit is open source, well documented and mailing-list is available for users query.

(b) Java

Java language offers several features that facilitate with easiness the development and deployment of a software environment for grid computing. As grid is network based Java's network based features are very useful to develop grid application. Java's network-centric approach and its built-in support for mobile code enable the distribution of computational tasks to different computer platforms. Java is an object oriented programming language. Java is used to manage the local information data of the server. A Java based platform potentially allows every computer in the Internet to be exploited for distributed, large scale computations; while at the same time the maintenance costs for the platform are minimal therefore Java has been used.

(d) MySQL

MySQL is used as the backend. MySQL is multithreaded, multi-user, SQL database management system. MySQL is the most used open source database. MySQL has become preferred choice for web based development for its speed, reliability and ease of use and hence MySQL is used in the implementation.

MySQL is the most popular open source database software, with its superior speed, reliability, and ease of use, MySQL has become the preferred choice for all because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in.

(e) PHP

PHP, which stands for "PHP: Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn.

The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but anyone can do much more with PHP.

(f) Apache HTTP server

Apache HTTP server is used to process and host the PHP web pages. Apache HTTP server takes PHP codes as input and creates web pages as output. Apache is open source and available for variety of operating systems. Apache has been used for hosting web pages on local network using Virtual Host function.

The combination of Apache HTTP server, MySQL, and PHP with Linux operating system creates a software bundle named LAMP. This software bundle is very popular among developers because of its low acquisition cost and ubiquity of its components and hence has been used.

### 4.4.2 Implementation of Algorithm

PHP has been used to create web pages through which users can search idle nodes. The web pages are used to monitor the nodes after submitting jobs to find out overloading. Three web pages have been designed.

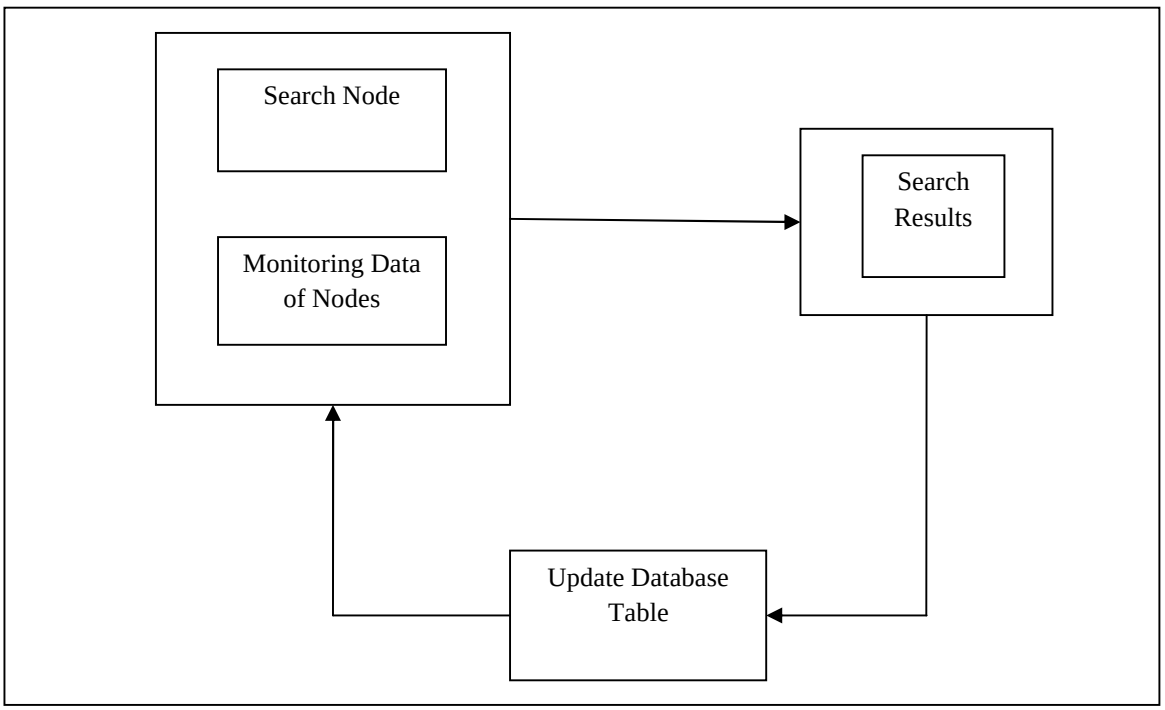


Figure 4.4 Design of Web Pages

These are searchnode.php, search.php and submitjob.php. searchnode.php provides a

form to search nodes compatible with jobs requirement. Search.php provides list of compatible nodes. If no compatible node found, then it generates an error message. submitjob.php updates database table which maintain job-node relation. searchnode.php and submitjob.php also provides monitoring data of nodes where jobs are executing. The implementation of algorithm has been shown with the help of this flowchart shown in above Figure 4.4.

The schema of jobSubmit table of mysql database is shown in Figure 4.5. This table has been used to track which job is submitted to which node and help to monitor the node respect to job also.

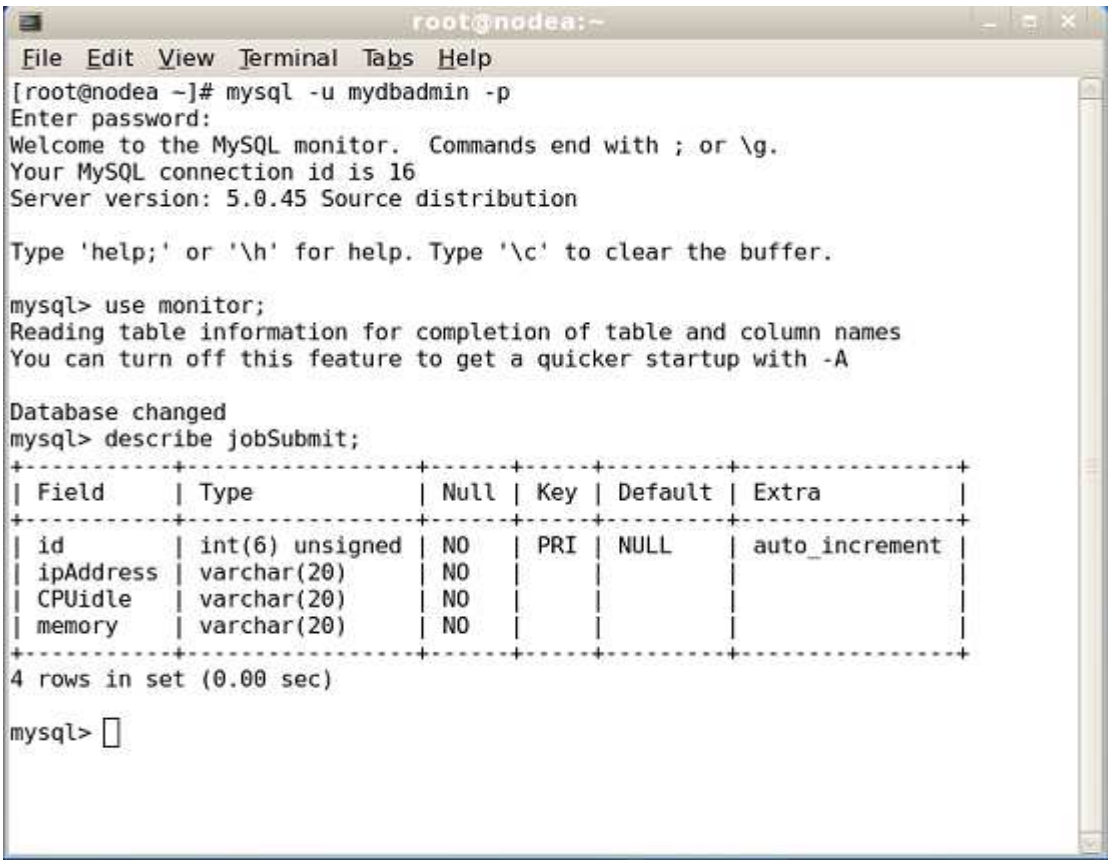


Figure 4.5 Schema of jobSubmit Table

## 4.5 Job Migration Technique

Job migration means re-allocation of jobs from one system to another system or in other words Job migration is a process of moving a normal, checkpointable or rerunable job from one host to another. In situations, where load balancing is required, job migration is

the only efficient way to guarantee that the submitted jobs are completed as per user requirements. In overloading situation chechpointed jobs or restartable job are migrated from overloaded system to underloaded system with the help of Gridftp.Gridftp is service that transfers the large amounts of data faster, more securely, and more reliably than FTP alone. GridFTP can be used either stand alone or as part of the Globus Toolkit [53].

## 4.6 Conclusion

This chapter discusses factors to initiate load balancing algorithms, comparative study of algorithms, used technologies and design and workflow of proposed algorithm. Next chapter discusses test results of proposed algorithm.

# Chapter 5 Experimental Results

This chapter discusses results of implemented algorithm in grid ennvironment.

## 5.1 Test Results

Figure 5.1 shows the main page.Users can search nodes according to CPU speed,idle CPU and free memory required by job. This page also provides monitoring data of all the connected nodes. The monitoring is shown per job and reflects, if any overloading found there.If the current idle CPU and free memory is less than that required by the job,it shows overloading depict in Figure 5.4.

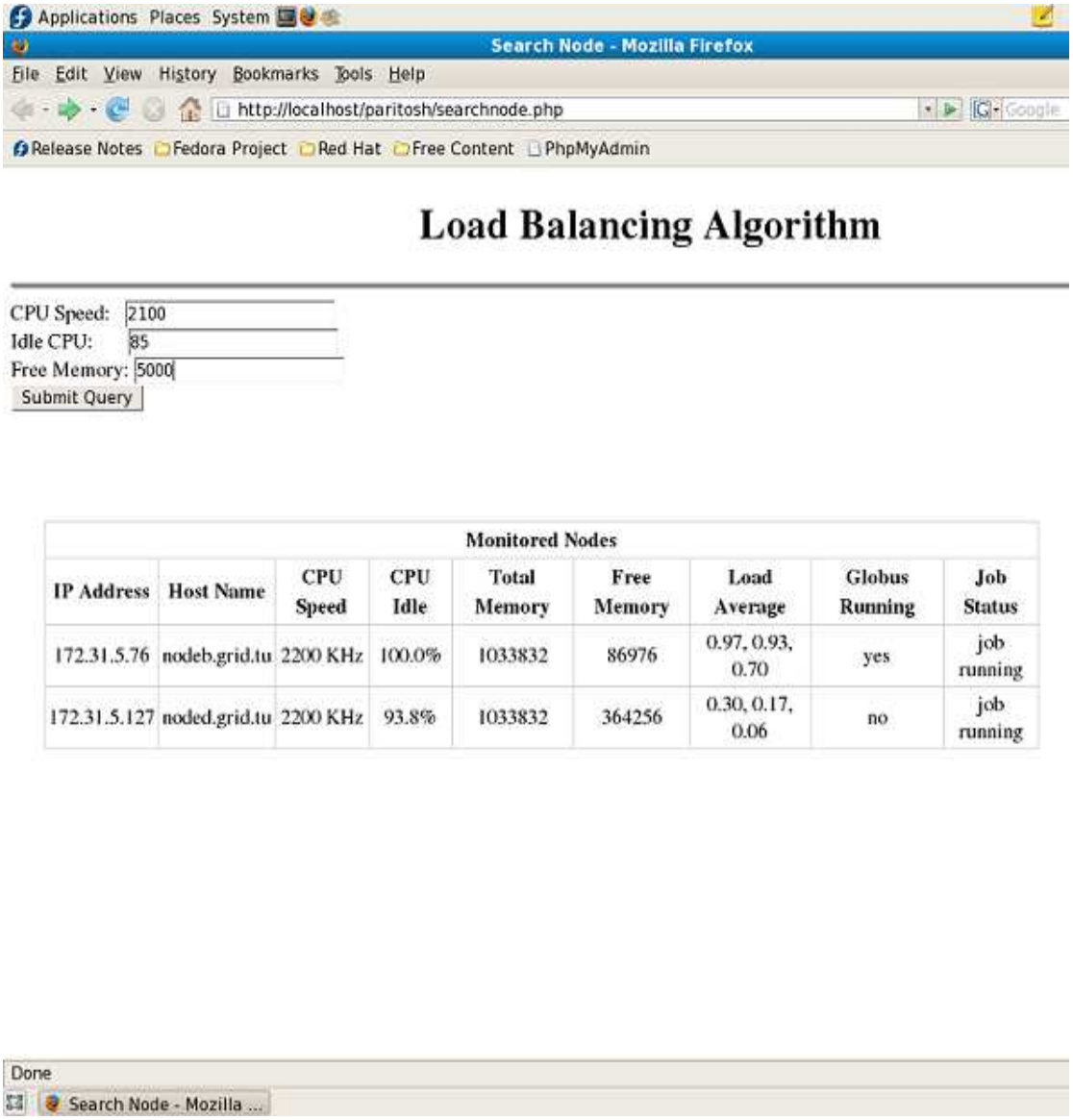


Figure 5.1 Main Page of Load Balancing Algorithm

Figure 5.2 shows the search results.It shows all the matched available nodes.It shows the IP address ,host name ,CPU speed,CPU idle,total memory,free memory and load average.It also shows whether the globus toolkit is running or not.It also provides the hyperlink to jobSubmit.php.

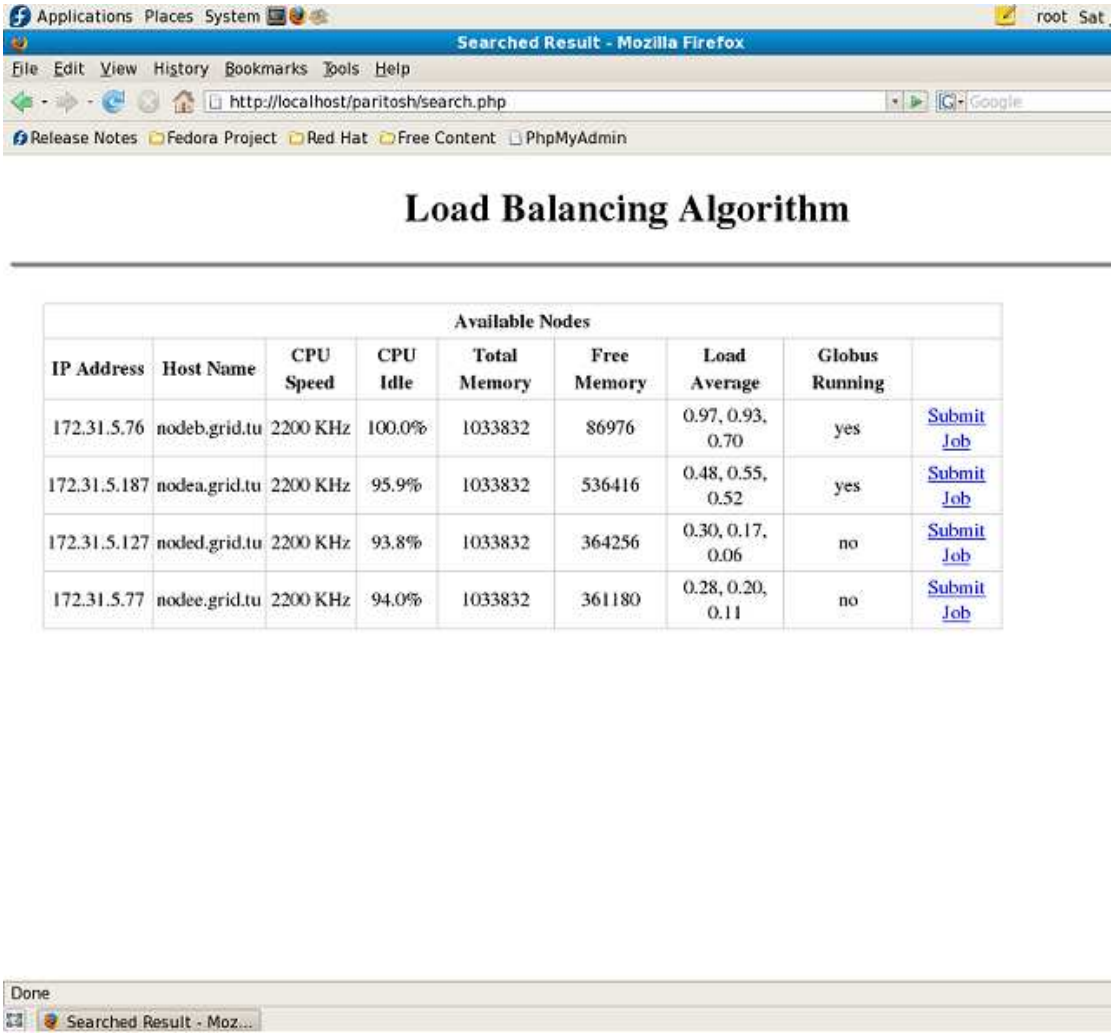


Figure 5.2 Status of Jobs

If no node is available with respect to search criteria than an error is shown as depicted in Figure 5.3. jobSubmit.php gets the required free memory ,idle CPU required by job and the IP address of the selected node from the search.php page.Then it updates the jobSubmit table of mysql database.This table is required to monitor the jobs status against nodes.This page redirects to the main page.

Figure 5.4 depicts a scenario where the available idle CPU and/or free memory is less than required by the job and the main page shows it by an overloading error. In this scenario two jobs are running at nodeb.One job is running and other is showing overloading because one job using resources according to requirement but another one require more than 30 % CPU idle,that's why overloading is shown.

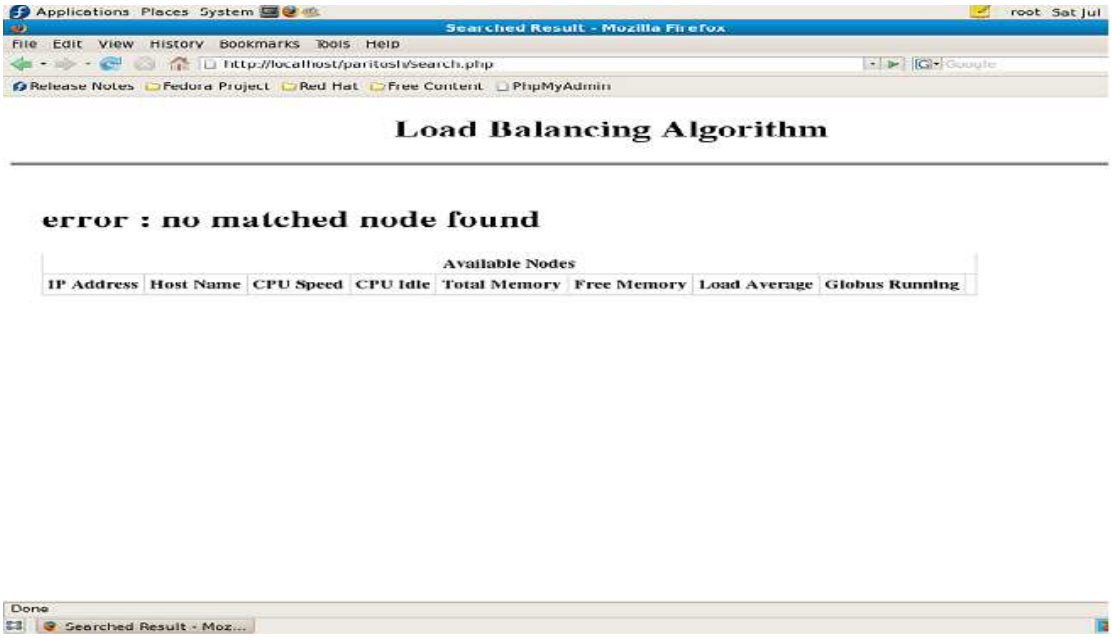


Figure 5.3 Error Messages

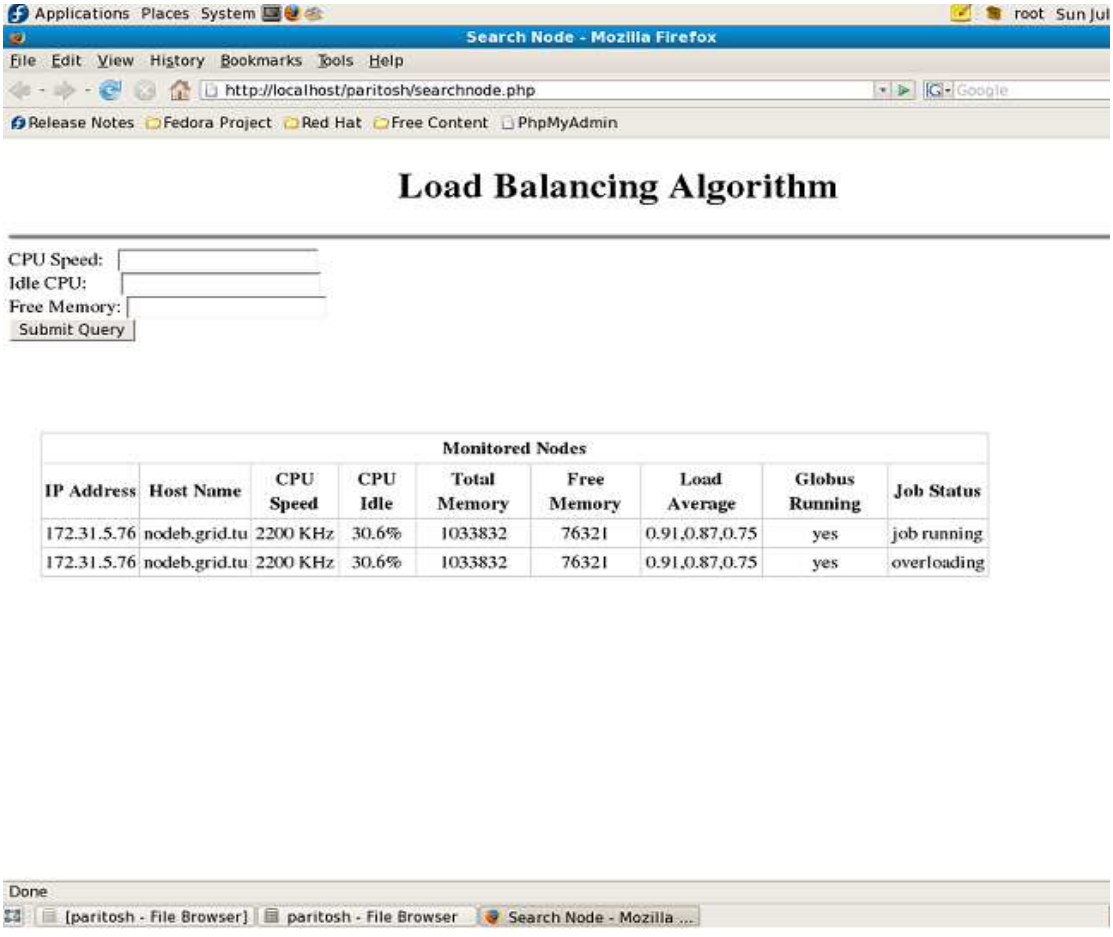


Figure 5.4 Overloading Message

## 5.2 Conclusion

This chapter shows the experimental results of proposed algorithm in different situations or modes.All the results best fit to expectation.Next chapter discusses the conclusion and future scope.

# Chapter 6 Conclusion and Future Scope

This chapter discusses the conclusions of the work presented in this thesis. The chapter ends with a discussion of the future direction which this work will take.

## 6.1 Conclusions

The work presented in this thesis describes multiple aspects of grid computing and introduces numerous concepts which illustrates its grand capabilities.Grid Computing is definitely a promising tendency to solve high demanding applications and its related problems.Main objective of the grid environment is to achieve high performance.

Dynamic nature and complexity of Grid make load balancing very complex and vulnerable to faults. To maintain entire load of nodes is very hard due to dynamic nature of resources in a Grid environment. There are a number of factors, which can affect the grid application performance like load balancing, heterogeneity of resources and resource sharing in the Grid environment.

This thesis focuses on load balancing and presents factors due to which load balancing is initiated, compares existing load balancing algorithms and finally proposes an efficient load balancing algorithm for Grid environment. It also presents Job Migration technique for balancing load in Grid environment.

## 6.2 Future Scope

This thesis work can further be extended by:

1.Migration of check-pointed jobs which would be restarted from last checkpoint.

2.Migration of jobs via grid-ftp instead of command-line.

3.Design and Testing of load balancing algorithm in a Multi-middleware scenario.

# References


[1]. Foster, I., Kesselman, C. and Tuecke, S. “The Anatomy of the Grid: Enabling Scalable Virtual Organizations” International Journal of High Performance Computing applications, Vol. 15, No. 3, pp. 200-222, 2001.

[2]. Klein rock, L. “MEMO on Grid Computing”, University of California, Los Angeles, 1969.

[3]. Jim Waldo, Geoff Wyant, Ann Wollrath, and Sam Kendall, “A Note on Distributed Computing”, Sun Microsystems Laboratories 2550 Garcia Avenue Mountain View, CA 94043.

[4]. David De Roure and others, “The Semantic Grid: Past, Present and Future”, Proceedings of the 2nd Annual European Semantic Web Conference (ESWC2005), Volume 93, Issue 3, 2005.

[5]. Ann Chervenak and others, “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets”, Journal of Network and Computer Applications, 2001.

[6]. Lord, P.Alper, P.Wroe, C. and Goble, C., “Feta: A lightweight Architecture for User Oriented Semantic Service Discovery”, In Proceedings of The Semantic Web: Research and Applications: Second European Semantic Web Conference (ESWC 2005), Heraklion, Crete.

[7]. Li, L. and Horrocks, I, “A Software Framework for Matchmaking Based on Semantic Web Technology”, In Proceedings of the 12th international conference on World Wide Web (WWW’03), Budapest, Hungary, May 2003.

[8]. Hai Zhuge “Special Section: Semantic Grid and Knowledge Grid” 17 July 2006; accepted 20 July 2006.

[9]. John Hagel, III and John Seely Brown, “Service Grids: The Missing Link in Web Services”, A Working Paper,

http://www.johnhagel.com/paper_servicegrid.pdf, 2002.(accessed: 18 September 2005).

[10].Mark Baker, Rajkumar Buyya, Domenico Laforenza "Grids and Grid Technologies for Wide-Area Distributed Computing" Software—Practice and Experience Soft. Pract. Exper. 2002; (in press) (DOI: 10.1002/spe.488).

[11].Daniel Adler "Grid Computing in Science" Georg-August Universidad Gottingen Institute for Informatics Software Engineering for Distributed Systems Group 08. November 2007.

[12].Ian Foster "What is the Grid? A Three Point Checklist (2002)" Argonne National Laboratory & University of Chicago July 20, 2002.

[13].Dazhang Gu, Lin Yang, Lonnie R. Welch "A Predictive Decentralized Load Balancing Approach" Center for Intelligent, Distributed and Dependable Systems, School of Electrical Engineering & Computer Science ,Ohio University.

[14].Javier Bustos Jimenez, Robin Hood "An Active Objects Load Balancing Mechanism for Intranet".

[15].Ian Foster, Carl Kesselman, and Steve Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of High-performance Computing Applications, 15 (3), pages: 200-222, 2001.

[16].History of Grid available at http://gridcafe.web.cern.

[17].Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez, "Grid Characteristics and Uses: a Grid Definition" Postproc. ofthe First European Across Grids Conference (ACG'03), Springer-Verlag LNCS 2970, pp. 291-298, Santiago de Compostela, Spain, Feb. 2004.

[18].Global Grid Forum, http://www.ggf.org.

[19].Lichen Zhang, “Scheduling Algorithm for Real-Time Applications in Grid Environment”, 2002 IEEE SMC TP1K5.

[20].Thilo Kielmann, Vrije Universiteit, Amsterdam, “Scalability in Grid”. PPT Core GRID, Bridging Global Computing with Grid (BIGG), Sophia Antipolis, France, Nov. 29,2006.

[21].Y.Lan, T.Yu (1995) “A Dynamic Central Scheduler Load-Balancing Mechanism”, Proc. 14th IEEE Conf. on Computers and Communication, Tokyo, Japan, pp.734-740.

[22].S.Rips “Load Balancing Support for Grid-enabled Applications” NIC Series, Vol. 33, ISBN 3-00- 017352-8, pp. 97-104, 2006.

[23].Belabbas Yagoubi and Yahya Slimani, “Dynamic Load Balancing Strategy for Grid Computing” Proceedings of World Academy of Science, Engineering and Technology Volume 13 May 2006 ISSN 1307-6884.

[24].Po-Cheng Chen,Cheng-I Lin,Sheng-Wei Huang,Jyh- Biau Chang, Ce-Kuen Shieh,Tyng-Yeu Liang, “A Performance Study of Virtual Machine Migration vs Thread Migration for Grid Systems”.

[25].S Kalaiselvi Supercomputer Education and Research Centre (SERC), Indian Institute of Science, Bangalore V Rajaraman Jawaharlal Nehru Centre for Advanced Scientific Research, Indian Institute of Science Campus, Bangalore “A Survey of Check-Pointing Algorithms for Parallel and Distributed Computers” vol. 25,Part 5,October 2000,pp.489±510.

[26].Parag Kulkarni & Indranil Sengupta Department of Computer Science & Engg. Indian Institute of Technology, Kharagpur, “Load Balancing With Multiple Token Policy” ICTA’07, April 12-14, Hammamet, Tunisia.

[27].P. Kokkinos, K. Christodoulopoulos, A. Kretsis, and E. Varvarigos Department of Computer Engineering and Informatics, University of

Patras, Greece and Research Academic Computer Technology Institute, Patras, Greece, “Data Consolidation: A Task Scheduling and Data Migration Technique for Grid Networks” Eighth IEEE International Symposium on Cluster Computing and the Grid.

[28].Srikumar Venugopal, Rajkumar Buyya and Ramamohanarao Kotagiri “A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing” Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia Email:fsrikumar, raj, raog@cs.mu.oz.au.

[29].Shahzad Malik, “Dynamic Load Balancing in a Network of Workstations”, 95.515F Research Report, November 29, 2000.

[30].Kai Lu, Riky Subrata and Albert Y. Zomaya, Networks & Systems Lab, School of Information Technologies, University of Sydney “An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites”.

[31].Guy Bernard,& Michel Simatic “A Decentralized and Efficient Algorithm for Load Sharing in Networks of Workstations”.

[32].Manish Arora and Sajal K. Das and Rupak Biswas “A Decentralized Scheduling and Load Balancing Algorithms for Heterogeneous Environment”.

[33].Gregor von laszewaski, Ian Foster, Argonne National Laboratory, “Designing Grid Based Problem Solving Environments”,www-unix.mcs.anl.gov/~laszewsk/papers/cogpse-final.pdf.

[34].Belabbas Yagoubi, Hadj Tayab Lillia and Halima Si Moussa Department of Computer Science, University of Oran Algeria “Load Balancing in Grid Computing” Asian Journal of Information Technology 5 (10):1095-1103,2006.

[35].Ferreira, L., Bieberstein, N., Berstis, V., Armstrong, J., "Introduction to Grid Computing with Globus," Redbook, IBM Corp., http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf.

[36].Foster, I., Kesselman C., Nick J.M. and Tuecke S. "Grid services for Distributed System Integration," Computer, vol. 35, no. 6, pp. 37–46, June 2002, http://www.globus.org/alliance/publications/papers/ieee-cs-2.pdf.

[37].http://www.globus.org/toolkit/docs/development/4.1.2.

[38].Francois Grey, Matti Heikkurinen, Rosy Mondardini, Robindra Prabhu, "Brief History of Grid",http://Gridcafe.web.cern.ch/Gridcafe/Gridhistory/history.html..

[39].Rafael A. Moreno "Job Scheduling and Resource Management Techniques in Dynamic Grid Environments".

[40].Yuan-Jin Wen; Sheng-De Wang Department of Electrical Engineering, Division of Computer Science, National Taiwan University, Taipei, Taiwan "Minimizing Migration on Grid Environments: An Experience on Sun Grid Engine" Journal of Information Technology and Applications vol. 1 No. 4 March, 2007, pp. 297-30.

[41].Huedo, E., Montero, R.S., Llorente, I.M.: "An Experimental Framework For Executing Applications in Dynamic Grid Environments" ICASE Technical Report (2002).

[42].Allen, G., Angulo, D., Foster, I., and others: "The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment. Journal of High-Performance Computing Applications", Volume 15, no. 4 (2001).

[43].Vadhiyar, S.S., Dongarra, J.J. "A Performance Oriented Migration Framework For The Grid" http://www.netlib.org/utk/people/JackDongarra/papers.htm (2002).

[44].K. A. Iskra and F. van der Linden and Z. W. Hendrikse and B. J. Overeinder and G. D.vanAlbada and P. M..ASloot, "The Implementation of Dynamite: An Environment for Migrating PVM Tasks", July 2000, ACM SIGOPS Operating Systems Review, Volume 34 Issue 3.

[45].Neogy, S. and Sinha, A. and Das, P.K., CCUML "A Checkpointing Protocol for Distributed System Processes", TENCON 2004. 2004 IEEE Region 10 Conference Volume B, 21-24 Nov. 2004 Page(s):553 - 556 vol. 2.

[46].Oliner, A.J. and Sahoo, R.K. and Moreira, J.E. and Gupta, M., "Performance Implications of Periodic Checkpointing on Large-Scale Cluster Systems", Parallel and Distributed Processing Symposium,2005. Proceedings. 19th IEEE International 4-8 April 2005 Page(s):8 pp.

[47].Jonathan M. Smith Computer Science Department Columbia University New York, NY 10027 "A Survey of Process Migration Mechanisms" Technical Report CUCS-324-88.

[48].M. Lizkow, M. Livny, and T. Tannenbaum, "Checkpoint and Migration of UNIX Processes in the Condor Distributed Environment", April 1997.

[49].Amnon Barak, Oren Laadan, and Yuval Yarom, editors, " The NOW MOSIX and its Preemptive Process Migration Scheme",TIEEE TCOS, 1995.

[50].A. Goscinski,"Research of Distributed and Open Systems and Processing: The RHODOS Project",Technical report, Deakin University, Australia, 1994.

[51].Fred Douglis. "Experience with Process Migration in Sprite". In Proceedings of the Symposium on Experiences with Distributed and

Multiprocessor Systems, pages 59{72, Berkeley, CA, 1989. USENIX Association.

[52].Eric Jul, Henry Levy, Norman Hutchinson, and Andrew Black. “Fine-Grained Mobility in the Emerald System”. ACM Transactions on Computer Systems, 6(1):109{133, February 1988.

[53].Peter Smith and Norman C. Hutchinson. “Heterogeneous Process Migration: The Tui system”. Software Practice and Experience, 28(6):611{639, 1998.

[54].María S. Pérez “Grid Computing: A Ten Years Look Back”,Facultad de Informática Universidad Politécnica de Madrid mperez@fi.upm.es.

[55].Dazhang Gu, Lin Yang, Lonnie R. Welch ,Center for Intelligent, Distributed and Dependable Systems ,School of Electrical Engineering & Computer Science ,Ohio University, “A Predictive, Decentralized Load Balancing Approach”.

[56].B. Yagoubi , Department of Computer Science, Faculty of Sciences, University of Oran and Y. Slimani , Department of Computer Science, Faculty of Sciences of Tunis, “Task Load Balancing Strategy for Grid Computing”.

[57].http://images.google.co.in/images.

[58].P.Senthil Kumar, S.Renuka Devi HOD Of MCA Department Cherraan’s Arts Science College, Kangayam. Research Scholar, Cherraan’s Arts Science College, Kangayam “ National Seminar on Bio Computing 2009”.

[59].Jorge R. Ramos Vernon Rego, Department of Computer Sciences Purdue University West Lafayette, IN 47907, U.S.A., Janche Sang, Department of Computer and Info. Science Cleveland State University Cleveland, OH 44139, U.S.A “An Improved Computational Algorithm for Round-Robin Service” Proceedings of the 2003 Winter Simulation Conference.

[60].Panos M. Pardalos, L.Pitsoulis1, T. Mavridou, and Mauricio G.C. Resende, “Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP” Center for Applied Optimization and Department of Industrial and Systems Engineering, University of Florida,USA AT&T Bell Laboratories,USA.

[61].Mathias Noack “Comparative Evaluation of Process Migration Algorithms” Dresden University of Technology Operating Systems Group 21st July 2003.

[62].Nalini Vasudevan, Prasanna Venkatesh “Design and Implementation of a Process Migration System for the Linux Environment”.

[63].William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, Ioan Raicu, Ian Foster, “The Globus Striped GridFTP Framework and Server” SC'05, ACM Press, 2005.

# Communicated Paper

1. Paritosh Kumar, Inderveer Chana "Comparative Study of Job Migration Algorithms for Autonomic Grid Management" *2nd International Conference on Computer and Electrical Engineering (ICCEE 2009)* 2009-01-01 - 2009-01-01.Dubai, UAE December 28 - 30, 2009