CHAPTER 3

FAULT-TOLERANT RECENT NEIGHBOUR LOAD BALANCING ALGORITHM

3.1 SYSTEM MODEL

Logically, the grid architecture is divided into three levels: Grid-Level, Cluster-Level and the Leaf-Nodes as shown in Figure 3.1. It is assumed that the Grid-Level consists of a collection of clusters connected by a communication network. Each cluster may contain multiple computing nodes called leaf nodes. The computing nodes in the cluster are heterogeneous in nature. The differences may be in the hardware architecture, operating systems, processing power and resource capacity. In this chapter, heterogeneity refers to the processing power of the computing node. The processing power of the grid cluster is measured by the average CPU speed across all the computing nodes within the grid.

Figure 3.1 Logical View of Decentralized Load Balancing Architecture
The communication overhead involved in capturing the load information of clusters before making a dispatching decision can be a major issue negating the advantages of the load balancing. Hence, the considerable dynamic communication delay in disseminating load updates is not ignored.

The clusters in the grid are fully interconnected meaning that there exists at least one communication path between any two clusters in the grid. Inter-cluster communication is achieved through message passing. There is a non-trivial communication delay on the communication network between the clusters. The communication delay is different between different pairs of clusters. The underlying network protocol guarantees that the messages sent across the network are received in the order sent (Kai Lu et al 2006).

For any cluster in the grid, there are jobs arriving at the cluster. The jobs are assumed to be computationally intensive, mutually independent and can be executed at any cluster. No deterministic or priori information about the job is available. As soon as the job arrives, it must be assigned exactly to one cluster for processing. When a job has been completed, the executing computing node returns the results to the originating computing node of the job. T is used to denote the set of all jobs, T= \{t_1,\ldots,t_k\}. Each job is assigned a timer when it is generated. If the timer reaches a threshold and the job is not processed, the job is given the highest priority for execution. The threshold for a job is computed only if the computing node to which the job was assigned fails. The unit for timer is in milliseconds. T-Master generates the timer and the threshold for each job. After each millisecond, the timer is incremented by one. Timer for each job is initialized to zero when the job is generated. The threshold is equal to the computing node to which the job would be re-assigned * the current timer of the job/computing node speed + the transfer delay to the computing node.
In a computing environment, the job migration is an efficient way to guarantee that the submitted jobs are completed reliably and efficiently when a computing node failure occurs (Malarvizhi and Rhymend Uthariaraj 2011).

### 3.1.1 Proposed Grid Model Components

Cluster-Level consists of a collection of computing nodes referred to as leaf nodes.

i. **Cluster Manager (CM):** It can fully control the leaf nodes within it, but cannot operate the leaf nodes of the other clusters directly. The leaf nodes within the cluster are referred to as neighbours and the leaf nodes of the other clusters are referred to as friends. Intra-cluster neighbours are formed in terms of the communication delay. A computing node \( n_j \) becomes a neighbor of a computing node \( n_i \) if the communication delay between \( n_i \) and \( n_j \) is within \( \alpha \) times the communication delay between the \( n_i \) and the nearest computing node say \( n_k \). In the experiments, it is found \( \alpha = 1.5 \) yields very good results and this value is used throughout the experiments. CM maintains the load information along with the registration information of its neighbours. In Cluster-Level, each leaf node runs a CM. CM’s role is to balance the intra-cluster workload.

ii. **Master:** A designated neighbour with the highest processing power in each cluster is treated as the cluster server called the master. For a given master, other connected masters are called partners. Master is responsible for accepting jobs from T-master. Inter-cluster partners for the grid are formed in terms of the communication delay. For a master say \( m_i \), a master \( m_j \)
is considered its partner as long as the communication delay between the master \( m_i \) and \( m_j \) is within \( \alpha \) times of the communication delay between the master \( m_i \) and the nearest master say \( m_k \).

iii. **Cluster System Monitor (CS):** Determines the load of the neighbours and provides this information to CM.

Grid-Level consists of a collection of interconnected clusters.

i. **Grid Manager (GM):** It is responsible for the load control among its clusters as shown in Figure 3.3. GM maintains the load information along with the registration information of each of the partners in the grid. The partner of each cluster also runs the GM as shown in Figure 3.2. \( N \) denotes the number of clusters in Grid-Level.

ii. **Grid System Monitor (GS):** Determines the load of the partners and provides this information to GM.

iii. **Client Interface:** Provides a graphical user interface to the user for submitting and receiving jobs.

iv. **Scheduler:** Responsible for the scheduling of the submitted jobs.

v. **Dispatcher:** Performs the dispatching of the jobs to other partners.

![Figure 3.2 Grid Topology](image)
vi. **Collector**: It is in charge of capturing the jobs from other partners.

vii. **T-Master**: A partner of any cluster in the grid called the T-Master is chosen for job generation and job assignment to other masters in the grid. T-Master also maintains the information about the jobs that have been assigned to each master of the grid. The leaf nodes of the cluster are responsible for completing the jobs assigned to them by their master. Scheduler runs as a sub-component of CM and GM.

viii. **Fault Detector (FD)**: Detects the failure and reports to the Cluster manager (CM).

ix. **Fault Notifier (FN)**: Detects the occurrence and type of the resource failure by analyzing the information about the state of a resource and transfers the information about the failure to FM.

x. **Fault Manager (FM)**: If FM receives the information about failure, it tries to resolve the failures. Fault manager guarantees that the jobs submitted are completely executed using the available resources.

Cluster System Monitor (CS) runs as a sub-component of CM. Grid System Monitor (GS) runs as a sub-component of GM. Dispatcher and Collector run as the sub-components of GM. Cluster manager maintains a local job-waiting queue which holds the jobs waiting to be assigned to intra-cluster nodes (i.e. neighbours) when load balancing is initiated. Grid-Level manager maintains a global job-waiting queue which holds those jobs waiting
to be assigned to inter-cluster nodes (i.e. partners) when load balancing is initiated.

RN uses run-time state information for decisions regarding the sharing of system load. All the masters of the grid system are involved in making the load balancing decision. RN method uses only partial information stored on the master to make a sub-optimal decision.

RN method does not use state-broadcast because the broadcast services are not available in grid systems. Instead RN method uses state-polling approach by setting a probe limit of a pre-defined interval between information exchanges. In a medium to heavily loaded system, if the probe limit is small, lightly loaded masters may not be discovered. If the probe limit is large, then most of the heavily loaded masters may find the same lightly loaded masters and load them. Repeated polling leads to wastage of processing time of the polling masters and polled masters. This problem is substantial if the grid system load is high. Repeated polling also generates a large amount of network traffic. This problem is major when the network bandwidth is limited. Also, as the jobs need to wait for the polling result, polling will increase the response time of the jobs. This is a problem if communication time is significant. Hence to overcome the problems caused by probe limit and repeated polling, the pre-defined interval is set to three times that of the job arrival rate. The pre-defined interval is modified when the job arrival rate changes.

In order to optimize the usage of processing speed and minimize the time required for load balancing decisions, RN method reduces the rate at which jobs are moved from one master to other. This is achieved by restricting the maximum number of jobs transmitted between the masters (i.e. a maximum of one job) at any given time.
Figure 3.3 RN Architecture
The jobs in the global and local job-waiting queue are processed in FCFS order. Jobs are scheduled based on immediate-mode scheduling method. That is, as soon as a job arrives at the master, based on the load information of itself and its partners, the job is assigned to a cluster which is least-loaded. Each Cluster manager maintains a node list NSET which contains information about the neighbours of the cluster and a cluster list CSET which contains the information about the partners of the grid. NSET and CSET are updated whenever a computing node enters or fails in the cluster.

3.2 JOB MODEL

For job \( t_i \) that belongs to a global job set \( T \) the following functions have been defined:

i. **BornNode**(\( t_i \)): denotes the originating computing node for the job \( t_i \).

ii. **ExeNode**(\( t_i \)): denotes the executing computing node for the job \( t_i \).

iii. **BornTime**(\( t_i \)): denotes the arrival time of job \( t_i \), which is the time when the job is generated at BornNode(\( t_i \)).

iv. **DeathTime**(\( t_i \)): denotes the finishing time of \( t_i \). It includes the job transmission time from BornNode(\( t_i \)) to ExeNode(\( t_i \)), waiting time queued at the ExeNode(\( t_i \)), processing times at the ExeNode(\( t_i \)), and the transmission time it takes to return the processing results from ExeNode(\( t_i \)) to BornNode(\( t_i \)).

v. **RespTime**(\( t_i \)): denotes the completion time of \( t_i \).

vi. **RespTime**(\( t_i \)) = DeathTime(\( t_i \)) − BornTime(\( t_i \)).
The objective is to have the minimum average response time for a set of jobs.

3.3 LOAD BALANCING MODEL

The RN algorithm is involved in getting the load information of the cluster and communicating this information to GM periodically which is later on used for the transfer of jobs among the clusters. Based on the load information RN chooses the most suitable node for each job and therefore, minimizes the job execution time and maximizes the system throughput.

Load information, generally defined in terms of the load is the necessary condition of RN load balancing algorithm. The load at each leaf node contributes to the overall load of the cluster and can be determined as the queue length. The load is determined dynamically and the weighted sum of squares method is used to calculate the load at each computing node.

Any cluster in the grid can be a processing cluster. In the cluster-level load balancing, depending on the current workload of its cluster, estimated from its own neighbours in NSET, each cluster manager CM decides whether to start a load balancing operation or not. If it decides to start a load balancing operation, then it tries to load balance the workload among its under-loaded neighbours in NSET. If any neighbour in the cluster at any instant of time is over-loaded, it allots jobs to other neighbours in NSET with a minimal load using the sender-initiated approach to load balancing. If the master is unable to load balance the workload among its leaf nodes, then the jobs are transferred to the under-loaded partners in CSET with a minimum load. Grid-Level load balancing is performed only if CM fails to load balance its workload among its leaf nodes.
In this case, the jobs of overloaded clusters are transferred to the under loaded ones in CSET according to the communication cost and the selection criteria. The selection criterion is based on the load of other partners. A cluster is said to be overloaded if its load is greater than the load of its partners. The chosen under loaded clusters are those which need the minimal communication cost for transferring jobs from overloaded clusters.

### 3.4 Fault-Tolerant Model

A fault detector is associated to each of the cluster managers and a fault notifier and fault manager to Grid manager. In case of a failure, the fault notifier, fault detector and fault manager work as follows:

Consider a computing node \( N_{ij} \) (\( i^{th} \) cluster, \( j^{th} \) node) that fails at a point of time:
Case 1: The failed neighbour is not a master of the cluster

1. The fault detectors associated with the neighbour nodes of $i^{th}$ cluster and fault notifier of $i^{th}$ cluster detect the $N_{ij}$ failure and the fault detectors report it to their cluster-level managers. The cluster managers update their NSET.

![Block Diagram of Proposed Fault Tolerant Model](image)

Figure 3.5 Block Diagram of Proposed Fault Tolerant Model

2. Fault notifier as shown in Figure 3.5 reports to the fault manager which decides whether to start a local jobs migration operation or not from $m_j$ to the rest of the neighbours of $N_{ij}$ in NSET. If it decides to start a job migration operation, then it load balances its workload among the neighbours in NSET. The decision is based on performance benefit of each job. The
performance benefit of a job aims to assign the job to a node which would benefit most in terms of the job’s response time.

3. If fault manager fails to migrate the jobs of $N_{ij}$ among its neighbours, then the jobs of $N_{ij}$ are transferred to the under loaded masters based on the communication delay. The chosen under-loaded masters are those which need a minimal communication cost for transferring the jobs from $N_{ij}$.

**Case 2:** The failed neighbour is a master of the cluster

1. The fault detectors associated with the neighbour nodes of $i^{th}$ cluster detect the $N_{ij}$ failure. The fault detectors report it to their cluster managers. The cluster-level manager updates its NSET.

2. The fault notifier of T-Master detects the $N_{ij}$ failure. The fault notifier reports it to the fault manager of T-Master.

3. The most powerful leaf node of the cluster becomes the new master of the cluster and other masters of the grid update their CSET.

**Case 3:** The failed neighbour is the T-Master of the cluster

1. T-Master failure is detected by other masters when they do not receive jobs after the arrival interval is over. The fault detectors associated with the masters detect the $N_{ij}$ failure. The fault detectors report it to their cluster managers. The cluster manager updates its NSET.
2. The most powerful leaf node of the cluster becomes the new master of the cluster and other masters of the grid update their CSET.

3. The under-loaded master of the grid becomes the T-Master. The T-Master requests other masters to have the current job assignment information.

3.5 PERFORMANCE STUDY

In this section, the performance of RN has been studied under different system parameters. It has been tested on a grid environment consisting of three clusters each containing two nodes. The communication delay was randomly generated within 10 time unit and assigned across the nodes. Then the nodes were sorted in ascending order based on communication delay. Node heterogeneity span is 5MIPS or 10 MIPS. The job heterogeneity span is 1 to 30 MIs. The transmission time for a job is assumed to be subject to the same time as the communication delay. The periodic load exchange interval is set to three times the job arrival rate. The node failure is set to 10 milliseconds. It is assumed that nodes do not fail simultaneously.

The performance of RN is compared with a method called ‘without RN’ which has no load balancing. In ‘without RN’ method, the cluster that generates jobs itself processes the jobs in FCFS fashion. The ‘Without RN’ method was developed for investigational purposes and to demonstrate the system performance when no load balancing and fault tolerance take place. For comparison with RN approach, a similar work is not found in the existing literature that takes into consideration load balancing and fault tolerance.
3.5.1 Average Response Time

The performance measurement of RN load balancing algorithm is based on five metrics: Total Execution Time, Mean Response Time, Communication delay, Number of Jobs and Number of Clusters. The mean communication delay between clusters is set to 5ms and job arrival interval is set to 1ms. The execution time of each job is equal to its waiting time at the processing node + the processing time on the processing node. The average response time is equal to average execution time + the average turnaround time. As the number of jobs increases the average response time also increases but the increase of the average response time is less in RN compared to the increase in the average response time for grid without RN as shown in Figure 3.6.

![Impact of RN on Average Response Time](image)

**Figure 3.6 Impact of RN on Average Response Time**

The average response time of RN increases as the communication delay increases as shown in Figure 3.7. The average response time of ‘Without RN’ remains same as it does not depend on the communication delay. Both the systems are heavily loaded i.e. 500 jobs. At a high system load, each computing node receives jobs more frequently which means each
computing node transfers jobs frequently. Hence the increase in the average response time of RN.

Figure 3.7  Average Response Time (ms) vs. Mean Communication Delay (ms)

3.5.2  Scalability

Figure 3.8  Average Response Time vs. Number of Clusters
As in Figure 3.8, the number of clusters is varied while keeping the system load constant. 400 jobs were generated by T-Master. The mean communication delay was set to 5ms. The mean initiation time of jobs was set to 1ms.

The system load can be varied by varying the mean initiation time of the jobs. The jobs were equally divided among the nodes of the cluster. Migration cost was set to zero if no jobs were transferred across the nodes. As shown in Figure 3.8, the average response time goes down as the number of clusters increases. However, the improvement comes at a decreasing rate. In moving from size 4 to size 6, the benefits of load balancing are very few or none. This suggests that particular sized clusters given their speed and bandwidth can be more effective with respect to load balancing for a given number of jobs with their computational requirements specified and job arrival rate given.

If more than one level of immediate neighbours is considered for job scheduling, the communication cost of jobs is increased leading to an increase in the job turnaround time.

3.6 SUMMARY

An efficient fault tolerant load balancing algorithm namely RN for intra-grid was presented, where the availability of jobs and resources is dynamic. Through simulation experiments, it is observed that the ‘Recent Neighbour’ algorithm provides a shorter response time, enhances the resource utilization and balances the load in an effective manner. The parameters measured are the average response time, system load and communication delay. The demerit of RN algorithm is that it periodically exchanges the load status information causing an extra overhead on the intra-grid performance.
With respect to fault-tolerance, the cluster-level managers become a bottle neck in gathering the failure information. When the cluster level managers themselves fail, the T-manager maintains a list of jobs that was sent to the cluster level manager for processing. T-manager also has information about the completed jobs. Hence, T-manager re-sends the failed jobs to an active cluster level manager once the failure of a cluster level manager is registered. An alternative approach to overcome the failure of T-Master is by having a dedicated backup T-Master that becomes active only when its primary T-Master fails. This comes with an assumption that both T-Masters will not fail simultaneously and incurs an additional replication overhead. The approach discussed in Chapter 4 overcomes the periodic exchange of load information demerit of RN by using the mutual information feedback policy.