Improving Load balancing by Flexible Assignment of Jobs in the Grid

[1]P.Neelakantan

[1]VNR VJIET,
Hyderabad -500090, India

***Abstract*: Grid computing has emerged as an attractive computing paradigm due to the availability of high speed wide area networks and low cost computational resources. This paper presents a novel load balancing algorithm for grid systems by considering node desirability. The set of partners and neighbours are formed for each node using the desirability of the job. For each job, in the gird, the proposed algorithm uses the desirability of other nodes in the grid to form k number of partners and p number of neighbours. The methods for building neighbours and partners are presented. A new job arriving to a node is immediately distributed to the originating node it or to its partner nodes. The load adjustment is carried continuously and reciprocal information management is used to minimize the communication overhead in the proposed load balancing algorithm. The proposed algorithm is dynamic, sender-initiated and decentralized.**

***Keywords*: Grid computing, load balancing, heterogeneous, job, wide area network, nearest neighbour.**

INTRODUCTION

The advances in computers and communications have changed society dramatically. At the same time, the computers can collaborate with the emerging high-speed networks by creating enormous computing power which helps in running advanced computational intensive jobs, in a minimal time.

There are different ways for setting up a distributed systems; cluster systems [5] and grid systems. Several personal computers or workstations are combined to form cluster systems which shall be used to run distributed applications through a high speed network. The disadvantage of using the cluster systems is, they are confined to a fixed area (e.g., [1, 2, 10]), making the job static in terms of its performance.

The geographically dispersed cluster systems connected by a network form computational grid for executing distributed jobs .As compared with the conventional clustered systems, grid computing uses internet connections to provide large scale resource sharing and improved resource utilisation. A Grid computing offers more processing power and quality of service in executing scientific jobs rather than the cluster systems. It also reduces the response time of the jobs. Computational grids will emerge as next generation computing and it will an important alternative for the computation problems in industry, academic and government organisations.

Resource management models: There are two kinds of resource management models and corresponding metrics.

System Centric: The grid consists of independent jobs which are submitted at different times and require different durations and resources for their execution. When a job arrives at a grid, the scheduler will analyse the load situation of every node and selects one node to run the job. The scheduling policy at this stage must optimise the total performance of the whole system. The scheduling system must realise the load balancing and increase the system throughput and resource utilization, if the grid system is heavily loaded. In this paper, this type of scheduling is classified as “system-centric scheduling”, for which the objective is to optimise system performance, such as [6], [7], [8], and [9]. The main focus here is the system –level resource.

Application Centric: The number of tasks of a parallel application arrives within a unit scheduling time –slot, the scheduling system will allocate a node and finish it in terms of a defined objective. Usually, the objective is the minimal completion time for the entire application. Here the scheduling policy is application oriented and hence it is referred an application centric scheduling. [2, 4, 9-11].

Application- centric models deal with three kinds of applications. . First is task producing, in which numerous independent jobs arrive simultaneously. The second is a co-allocation application, where each task is modelled as performing all-to-all communication patterns throughout its execution. The last kind of application can be represented as direct acyclic graph (DAG) indicating data-dependency between the tasks.

Considerable transfer cost.

The transfer cost of remote job execution at the local area network can be ignored because the computers in the LAN are connected through a high speed network. However, the transfer cost is a concern for the scheduling algorithm to execute the job in the remote system in the grid, due to the low speed Internet links.

Uneven job arrival pattern: The utilization of computers exceeds the maximum capacity at peak intervals and drops to a minimum in the night hours. A bursty traffic generated by the nodes in the grid environment will be balanced by distributing the workloads to different clusters. Hence, load balancing optimises the resource usage and designing a load balancing algorithm in a grid environment is more complex. The main motivation of this study is to propose, decentralized dynamic load balancing solutions that can cater to these unique characteristic of Grid computing environment

Literature Review

There exists many load balancing algorithms in the literature. Classification of the load balancing algorithms is useful for the design and analysis of new load balancing algorithms.

Static versus dynamic

Load balancing operations shall be carried out at compile time or during run time. Load balancing algorithms may require priori information about all the characteristics of the jobs,

nodes processing capacity and the communication speed between the nodes. Deterministic or probabilistic load balancing decisions are made during compile time and those decisions remain constant during run time.
The static load balancing algorithms offer less run time overhead and ~~it is~~ are simple. However they do not consider dynamic run time environment and may lead to load imbalance on some nodes which significantly increases the job response time. The nodes in the cluster environment exhibit dynamic load behaviour varied with time and require dynamic scheduling decisions based on the load value of the nodes. The dynamic load balancing algorithms uses run time state information to share load among the nodes in the system. The system performance is increased by distributing the load among the by better responding to the current system state.
The information collection during run time poses communication overhead and is a disadvantage of the dynamic load balancing algorithm. Though the time complexity of the dynamic load balancing is higher than the static load balancing algorithm, they provide better system utilization. However hybrid algorithms use the advantages of both static and dynamic strategies. The hybrid algorithm first uses the static algorithm for a "coarse " adjustment and then the dynamic load balancing algorithm is applied for "fine" adjustment by moving the jobs in the queues to the other lightly loaded nodes in the system. The algorithm proposed in this paper belongs to hybrid class.

Non-pre-emptive versus pre-emptive

Dynamic load balancing policies may be pre-emptive or non-pre-emptive. A non-pre-emptive load balancing policy [8, 9, 3,4,5] assigns a new job to a best node in the system. Once the job execution begins at that node, it is not moved even if the node run time characteristics change. However it is desirable that the load at each node need not be fully equalized. This property allows to devise load balancing schemes that deal with large grain division of the workload such as tasks and it does not require high speed communication between nodes.
Non pre-emptive load balancing policies can be suited for loosely coupled systems and it can be applied to any type of distributed system with homogenous or heterogeneous computing nodes such as a grid system.
In contrast, a pre-emptive load balancing will perform load balancing among the nodes whenever an imbalance exists among the nodes. A job can be transferred to another node even if the job is in its course of execution. Initially, though load distributions across nodes appear to be balanced, they will become unbalanced as shorter jobs complete and leave behind an uneven distribution of longer jobs. Job migration allows these imbalances to be corrected among the nodes. However, job migration in course of execution incurs more overhead which results in performance degradation. If the pre-emptive policies are attempted in loosely coupled systems, more messages are to be generated among the nodes and it will cause congestion in the communication system which will result in the system performance degradation
Many studies have shown that job migration is difficult in practice [11, 9, 10]) and the operation is more expensive and no significant benefits over non migration. Hence, this paper considers only non-preemptive load balancing strategies.

2.1.3 Node-level versus grid-level

When a job arrives at a node, the load –balancing algorithm of the cluster will analyse the load situation of every node in the cluster and will select an appropriate node to run the job. Even if the cluster is heavily loaded, each job in the cluster must queue and wait for its turn. This kind of load-balancing is classified as cluster-level load balancing which optimizes the system performance in a single cluster. Many traditional load balancing algorithms fall in the category of cluster-level [8, 9, 11].
On the contrary, if a cluster lacks sufficient resources to complete the newly arriving jobs or the cluster is heavily loaded, the load balancing system of the cluster will transfer some jobs to other clusters and will increase the system throughput and resource utilization in multiple clusters. This load balancing is referred as grid-level load balancing. The focus of this paper is on grid level load balancing [8, 11, 10, 6]

2.1.4 Centralised versus distributed

Load balancing policies can be classified as centralized or distributed. In centralized policies, there will be one master node which will take decisions about scheduling. The master node assigns newly arriving jobs to different processing nodes. The information collection about job arrivals and departures will be easy in the centralized policies. The major disadvantage of centralized policies is the possible performance and reliability bottleneck due to the possible heavy load on the master node. For this reason, centralized policies are inappropriate for large scale systems.
However, on other hand, the distributed policies involve all the individual nodes as decision makers. Jobs arrive at each node in the cluster and decisions will be made based on the partial or global information available at the node. The distribution policy is referred as individual optimal policy, in that each job optimizes its own response time independently of others. [10, 3].

System Model

The system model for the load balancing algorithms is composed of a (i) Cluster model (ii)Job queue model (iii) communication model (iv)Job model (v) Job migration model and (vi)Performance objective. The Grid architecture consists of a collection of clustered systems and job queue model provided as a two level architecture for the job waiting queue at each cluster. The communication model provides an estimate of expected communication costs for information exchange between nodes and job transfer among cluster nodes. The job model defines the information about the job required by the load balancing algorithms. The load balancing algorithm is designed in such a way that it has to reduce the chances of job thrashing and starvation at any node. The performance objective of the load balancing algorithm will be system utilization and average response time of the jobs.

Architecture model

The clusters consist of N number of processors and the communication bandwidth is shared by all the processors. The previous research such as condor[3][5] and Load sharing facility has addressed the management of the jobs at cluster level.
The heterogeneity in system can be expressed in terms of processors speed, memory and disk I/O. A practical solution is to consider CPU speed. It is also assumed that a machine with powerful CPU will have matching memory and I/O resources. The nodes in the grid system may have different processing power. Processing power of a node n_i is denoted as P_i. For

$i \neq j$, P_i may be different from P_j. P_i means the ratio of the processing power of a node n_i to the processing power of the slowest side s_j in the system.

Communication model

The nodes ***N*** are fully interconnected and there exists at least one communication path between every two nodes in N. The message passing mechanism is used as a communication between the nodes and there exists a transfer delay on the communication network between the nodes. The transfer delay is different between different pairs of nodes. The underlying network protocol guarantees that messages sent across the network are received in the order sent.

Two parameters such as a transfer delay and data transmission rate are used to represent the network performance between any two nodes (n_i,n_j). Transfer time required for sending a message of Q bytes between two nodes is given by

$$\mathrm{TD}_{ij} + \frac{Q}{\mathrm{BW}_{ij}}$$

The above equation represents the total time required to traverse all of the links on the path between n_i andn_j. BW_{ij} is represented as effective data transferring rate in bytes per unit time or is characterized in terms of kb/s. TD_{ij} includes a start-up cost and delays incurred by congestion at intermediate links on the path between n_i and n_j.

For a given node $n_i \in$N, jobs will arrive to the nodes belong to C_i , where Ci denotes the cluster consisting of N nodes. The arrival of jobs are random with an average delay,λ, between two successive arrivals and follow poisson rate and the delay between the arrivals will be Exponential distribution. The jobs can be executed at any node and are computationally intensive. The execution of the jobs are not time shared and can be executed at any single node. The job is assigned to exactly one node for execution and on completion of job, the results will be transferred to the originating node of the job. The set of all jobs generated at node S will be denoted as J= $\{\{j_1, j_2 \ldots j_r\}\}$. The system automatically creates the following parameters related to the job.

- orgNode(j_i) : the originating node of job j_i
- exeNode (j_i) : the executing node of the jobj_i
- startTime (j_i) : the time of the job generated at orgNode (j_i) which is the arrival time of the job.
- endTime(j_i) : the completion time of j_i which includes the job transfer time from orgNode to exeNode(j_i) ,waiting time queued at the exeNode(j_i), execution times at the exeNode(j_i) and the transfer time it takes to return the execution results from exeNode(j_i) to bornNode(j_i)
- respTime (j_i):the time the job ji taken for execution. respTime (j_i) =endTime (j_i) −startTime (j_i).

Each job j_x is represented by two parameters, the amount of computation and the amount of transfer time. The unknown values of these two parameters may be estimated by probabilistic techniques. The amount of computation has one of the following formats.

An Expected execution time ETC(j_x, N_{std}) for processing the job, is the time that would be taken at a standard platform with processor speed equal to 1 . Hence the expected execution time of a job ETC (j_x, n_i) will be $\frac{\mathrm{ETC}(j_x, N_{std})}{\mathrm{APW}_i}$

Performance objective

The major critical performance object in the grid computing is to minimize the response time of all jobs submitted in the system denoted by ART.

$$\mathrm{ART} = \frac{\sum_{i=1}^{n} responsetime\ (j_i)}{n}$$

The performance of the proposed algorithm is evaluated by its improvement factor over the another algorithm X as follows in terms of average response time of jobs

$\frac{\mathrm{ART}(X) - \mathrm{ART}(A)}{\mathrm{ART}(X)}$, Where ART (A) denotes the average response time of jobs using algorithm A and ART(X) denotes the average response time of jobs using algorithm X. A positive value indicates an improvement over the existing algorithm and negative value implies the degradation over the existing algorithm.

PROPOSED METHOD

A novel load balancing algorithm for heterogeneous systems has been presented in this paper with consideration of job migration to the remote nodes. Job migration from a local node to remote node considers processing power of a remote node and the communication delay to the remote node. The load balancing algorithm for each node n_i forms a set of K partners and C neighbours and the information collection overhead from the neighbour and partner nodes are reduced by using reciprocal information management (RIM). The algorithm presented in this paper is dynamic, sender-initiated and decentralized. Job that arrives at each node n_i is assigned either to n_i or to its neighbouring nodes. The adjustment of loads has been made continuously among neighbours of node n_i.

Resource -aware load-balancing algorithm(RWLB)

Many existing algorithms in the literature have used an instantaneous run queue length (the number of jobs being served or waiting for service at a given instant) as the load index [11, 10]. The time required for calculating the load index is based on the queue length of the node. The load index of a node consisting of more than one CPU is calculated based on the total queue length of that node divided by the number of CPUs at that node. The parameters such as average processing power and the transfer delay are used to assign a job to a node in the node.

The node- clustering algorithm considers N number of nodes for the processing power of each node n_i. The nodes are chosen randomly in such a way that the processing power of each node varies large enough with other node. The nodes are sorted by processing power in descending order before applying node-clustering method. A reference vector $< d_1, d_2, d_3 .. d_n >$ is calculated based on the difference in processing power Pi of node n_i to the other nodes in the system and nodes with similar reference vector close to each other in terms of processing power are clustered into $c_1, c_2, c_3 .. c_m$ clusters. Finally empty clusters in $c_1, c_2, c_3 .. c_n$ are removed so that only $c_1, c_2, c_3 .. c_q$ $(q \leq m)$ will remain in the order of decreasing average processing power.

The probability of false clustering will be reduced by selecting sufficient reference nodes that have very different processing power. This approach belongs to coarse grained approximation the proposed load balancing algorithm does not require precise measurements. The clusters generated are then used to generate partner nodes.

Partners

Each node n_i has K number of partner nodes P_{set} used by the scheduler to select partner nodes for assigning newly arrived jobs. When a node joins the grid system, it will determine its partners. A simple heuristic is employed to find partner nodes including heterogeneous nodes in terms of their computing power. An algorithm to select partners for the nodes is presented in the below algorithm.

A preferable collection of nodes of Nof forms a set Q_i used in proposed Partners Adjustment Policy have greater or comparable processing power to node n_i .The set of favoured nodes Q_i will be updated by the algorithm as necessary. The above approach may not guarantee in finding the optimal partners, however it may provide a scalable and efficient approach in the initial formation of partner nodes.

A. $Algorithm1: FindPartners\,(n_i, K)$

```
Find all nodes $n_j \in N(i \neq j)$ with $p(n_j) \leq P(n_i)$. The set of nodes are denoted as $Q_i$.
if$\beta$>=K ($\beta$ is the size of $Q_i$)
    select K nodes from $Q_i$ randomly and add them to $P_{seti}$
else
{
M= K- $\beta$;
Add $Q_i$ to $P_{seti}$
$v \leftarrow P(n_i)$
do
    {
    $Q_i \leftarrow Q_i \cup C_v$
            if $\bar{\beta}$>=M($\bar{\beta}$ is the size of $C_v$)
            {
            Select M nodes from $C_v$ randomly and add them to $P_{seti}$ ;
            break;
            }
            else
            {
            Add $C_v$ to $P_{seti}$
            M=M-$\bar{\beta}$
            $v = v + 1;$
            }
    } While M>0

}
```

Neighbours

The set N_{seti} maintains C number of neighbouring nodes by each node n_i . The scheduler will reduce the communication cost by selecting the neighbouring nodes for migrating the jobs and hence reduces the transfer delay for the load movement and enable quick response to load imbalances. The neighbours have been selected in such a way that nodes are lightly loaded and minimum transfer delay between the sender node and the receiver node. n_j is considered as neighbouring node to n_i if the communication delay between the nodes n_i and n_j is minimum. The neighbouring nodes are sorted in the ascending order based on the transfer delay and the least ranked node is chosen as the nearest node. The transfer delay is described as follows:

$$\varepsilon = \frac{\mathrm{TD_{ji}}}{\mathrm{TD_{near}}}$$

The transfer delay from node n_j to node n_i is denoted by TD_{ji}. The transfer delay from the nearest node of n_i is denoted by TD_{near}.

Partners Adjustment Policy

When a node receives a load information message from neighbour nodes or partner nodes, it triggers dynamic partner adjustment policy. The dynamic Partners Adjustment Policy is triggered whenever a node n_i receives load information message from a neighbour or partner. If a node n_j in the preferred nodes Q_i of n_i is found in the message, it will be involved in the partner adjustment of n_i. when noden_j load is lower than highest load in the partner nodes of n_i, then it is possible that n_j becomes a partner node of node n_i.Algorithm 2 describes the procedure of Partners. Adjustment Policy when n_i receives an information message from its neighbour or partner node n_i

B. Algorithm 2 Partners Adjustment ($\boldsymbol{n_j}, \boldsymbol{K}$)

```
{
$N_I \leftarrow \emptyset$;
$\forall\, n_y \in N$: if($n_y \notin Nset_j \cup P_{setj}$) and ($n_y \in Q_j$) $N_1 \longleftarrow N_1 \cup n_y$
if $N_1 \neq \emptyset$
{
$N_1 \leftarrow N_1 \cup Pset_j$
Sort $N_I$ by load difference in ascending order
Remove all nodes from $P_{setj}$
Select the first k nodes from $N_1$ and add them to $P_{setj}$
}
```

Information policy

The reciprocal information management system restricts the load information exchange to partner and neighbouring nodes ton_i. When a node n_i transfer a job j_x to its neighbour or partner node n_j for processing ,. Node n_i appends load information to itself ,r_p , random neighbours or partners who have sent the job transfer request TR to n_j . The load information is updated by n_j by comparing the timestamp, by inspecting whether a request is from it is neighbours or partners. Similarly n_j inserts its current load information and r_p radon nodes from its N_{seti} and P_{seti} in the job acknowledge AR or completions reply CR to n_i, So n_i can update its state objects.

For any node $n_i \in N$ if the state object element $O_i[j]$($\forall n_j \in$ Nseti$\cup$ pseti,i≠j has not been updated for a predefined period T_p , then the load-balancing scheduler will send information exchange message ton_j. The procedure is the same as the algorithm 4.3.

The message overhead in the periodic information exchange is reduced by using RIF method. In this method, the node under processing will return its current load and the load of r_p random nodes along with ACK message or completion reply CR back to the forwarding node. Hence the overhead is minimal. Another advantage of the RIM method is the rate of load dissemination is directly related to the job arrival rate. The load information is exchanged often where there is more number of job requests. The load information exchange rate is automatically adjusted to the request rate.

C. Algorithm 3:Instantaneous Distribution Policy

```
$\forall j_x \in J$ with bornnode$(j_x) = n_i \in N$
Let $LD_{Min} \leftarrow Min\{O_i[K].LD | n_k \in n_i + P_{seti}\}$
/* the minimal load among node $n_i$ and its $P_{seti}$ */
If $O_i[i].LD - LD_{Min} < \theta$)
        /* $\theta$ is a positive real constant close to 0 */
$GJQ(n_i) \leftarrow enqueue(j_x)$ /* put the job $j_x$ in the global job queue GJQ $(n_i)$ */
else
{
Transfer the job$j_x$ to the partner node $n_j$ having $LD_{Min}$
Update $O_i[j].LD$
}
```

Transfer policy and location policy

The transfer and location policies used in the proposed algorithm are combination of instantaneous and load adjustment policy.

Instantaneous Distribution Policy

The instantaneous distribution policy is used to decide whether a new job is assigned to the originating node or one of its neighbour nodes. If the existing neighbour nodes of n_i are overloaded then the job is put in the global queue of n_i which is later scheduled to run on the partner nodes. This policy will try to control the job processing rate at each node and highly computational jobs are run on the high end nodes or very less overloaded nodes. If there are two partner nodes with the same minimum load, the nearest partner node is chosen for executing the job and this can reduce the communication delay. Algorithm 4.4 describes the IDP for n_i

D. Algorithm 4: Load adjustment Policy

```
if $O_i[i].LD > LDavg$
{
$j_x \leftarrow dequeue(GJQ(n_i))$
Transfer the job $j_x$ to a neighbour node $n_j$ where $O_i[i].LD = Min\{O_i[k].LD | n_k \in Nset_i\}$
}
```

Load Adjustment Policy

The Load adjustment policy reduces the load difference between node i and its neighbour nodes by transferring jobs from heavily loaded nodes to the lightly loaded neighbouring nodes. This policy is triggered when the load information is received by node I from its neighbouring nodes. This policy uses the most recent load information to decide to initiate the transfer of jobs. The threshold policy used in this method is dynamically adjusted based on the system load and the nodes with loads higher than the average load of the system are considered as senders and the last job waiting in the node I is considered as the candidate for transfer to the remote node. If the neighbouring nodes have the same minimum load, then the candidate node for transferring the job is chosen based on the network delay. The node with less transfer delay as considered for transferring the job.

The Load Adjustment Policy for a node n_i tries to continuously reduce load difference among n_i and its neighbours n_{seti} by transferring jobs from heavily loaded nodes to lightly loaded neighbouring nodes. The policy is triggered whenever n_i receives load information from a neighbour. The policy will use the most recent load status

RESULTS AND DISCUSSION

In this paper, Sender-Initiated algorithms are used for performance evaluation. The proposed algorithm (RWLA) is compared with two of the existing algorithms in the literature. For the NB, each node is limited to collect load information from within its own domain, which consists of itself and its neighbours. The load balancing action is initiated if a load of node exceeds the average load of its domain.

Simulation model

Simulations are preformed to study the performance of the proposed algorithm by comparing with the existing algorithm diffusion Algorithm existing in the literature Robert Elsässer(2000). The simulation model consists of N nodes with processing power of nodes is assigned in a range [0.0 to 1.0].It is possible to produce different heterogeneous systems by varying the processing power of the nodes, it is possible to produce different heterogeneous systems. Jobs arrive at each node n_i,i=1, 2..., N according to a Poisson process with rate $\lambda_i = \lambda X P_i$ where $p_i = \frac{1}{N}$. The actual inter arrival time of jobs is adjusted to average system loading. The execution times of jobs are assumed to be an exponential distribution with a mean of T time units. The transfer delay between any two nodes assumed to be lognormal distribution with a mean of τ time unit and a standard deviationσ_c. The partner set of each node is provided in prior to the stating of the algorithm. The neighbours are chosen based on the transfer delay generated from the distribution of mean transfer delay and it has been provided as input to the algorithm before it starts to run. The average load of the system is denoted by ρ and defined as the ratio of average job arrive rate divided by the average job processing rate. The mean inter-arrival time $\frac{1}{\lambda}$ to get the desired value of ρ . Table 1 shows the simulation parameters used in the experiment and table 2 shows the heterogeneous system configurations used in simulation.

Simulation parameter	Value
Size of system, n	16
The number of reference nodes, m	6
Mean processing time of jobs	0.5 Time units
Computation to communication ratio	{0.05,0.1,0.25,0.5,1, 2.5,5}
Mean transfer delay	0.0025 Time units
Standard deviation of transfer delay	25
Period for periodic information exchange	20
Number of random partners/neighbours for information update,r_p	4

Table 1: Simulation parameters

The below table shows heterogeneous system configurations, and the value is chosen randomly from [0.0 to 1.0]. The first 2000 jobs are used to make the system into a steady state. The arrival processing time and finish time has been traced from J_{1000} To J_{4999} . Here μ equals to 4000 (for evaluation purpose).

For each node, the numbers of completed jobs are recorded. The computed average response time of jobs (ART) has been measured after each simulation run. The measurement has been carried out five times with different random seeds.

Heterogeneous systems	Average processing power
1 To 8 nodes	0.1
9 to 16 nodes	0.2
17 to 24 nodes	0.5
25 to 32 nodes	1.0

Table 2: Heterogeneous system configurations

Effect of system heterogeneity

The simulations have been carried out for four different heterogeneous systems under different system utilization parameter ρ. In the beginning of the simulation, the fastest nodes that have 10 times higher relative processing power than the slowest node have been considered. The system load is varied by varying the mean inter arrival time of the jobs, $\frac{1}{\lambda}$ and results are shown in Figure 4.1.

Effect of system size

The simulations have been carried out for varying system sizes to check for the stability of the proposed and existing algorithms. Both algorithms are scalable and stable. However the average response time offered by the proposed algorithm is better than the diffusive load balancing algorithm. The results are shown in the figure 4.2

CONCLUSION

The resource aware dynamic load balancing algorithm (RWLB) proposed in this paper by considering the scalability of the grid system which consists of heterogeneous processing nodes and addressed considerable communication overhead involved in collecting the information from the various nodes in the grid. The simulation revealed the performance of the proposed load balancing algorithm compared with the existing diffusion load balancing algorithm and the experimental results show that the proposed algorithm has done better than the diffusion load balancing algorithm and reduces the average response time of the jobs.

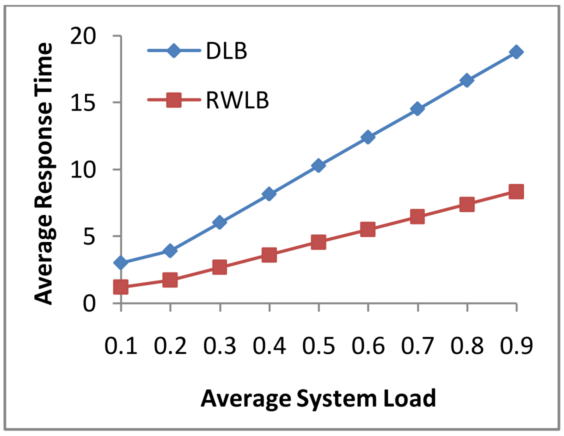

Figure 4.1: Effect of System heterogeneity

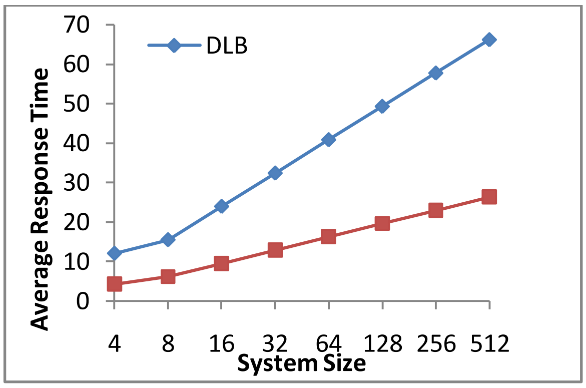

Figure 4.2: Effect of System Size

REFERENCES

[1] S. F. El-Zogdhy, H. Kameda, and J. Li, Numerical studies on a paradox for noncooperative static load balancing in distributed computer systems, Computers andOperations Research, 33(2) (2006) 345-355.

[2] S. Penmatsa, A.T. Chronopoulos, Cooperative load balancing for a network of heterogeneous computers, in: Proceedings of the 20th IEEE International Parallel and Distributed processing Symposium, 25-29 April 2006 Page(s):8.

[3] Z. Zeng and B. Veeravalli, Design and analysis of a non-preemptive decentralized load balancing algorithm for multi-class jobs in distributed networks, Computer Communications, 27(7) (2004) 679-694.

[4] K. Lu, R. Subrata, and A. Y. Zomaya, An efficient load balancing algorithm for heterogeneous grid systems considering desirability of grid sites, in: Proceedings of the 25th IEEE International Conference on Performance, Computing, and Communications, 10–12 April 2006, Phoenix, Arizona, USA.

[5] K. Lu, R. Subrata, and A. Y. Zomaya, Towards decentralized load balancing in a computational grid environment, in: Proceedings of the first International Conference on Grid and Pervasive Computing, May 3-5, 2006, Taichung, Taiwan, Lecture Notes in Computer Science (LNCS), Vol. 3947, pp. 466- 477, Springer-Verlag Press.

[6] K. Lu and A. Y. Zomaya, A hybrid policy for job scheduling and load balancing in heterogeneous computational grids, in: Proceedings of the 6th IEEE International Symposium on Parallel and Distributed Computing, 5-8 July 2007, Hagenberg, Austria.

[7] Jia Z. A Heuristic clustering-based task deployment approach for load balancing using Bayes Theorem in cloud environment. IEEE Transactions on Parallel and Distributed Systems. 2016; 27(2):305–316.

[8] Kunjal G, Goswami N, Maheta ND. A performance analysis of load Balancing algorithms in Cloud environment. 2015 International Conference on Computer Communication and Informatics (ICCCI), IEEE; 2015. p. 4–9.

[9] Ashok, A.S., Hari, D.P., 2012. Grid Computing: various job scheduling strategies, emerging trends in computer science and information technology. In: Proceedings of the International Conference on Emerging Trends in Computer Science and Information Technology-2012 (ETCSIT-2012), Mar.

[10] Ruay-Shiung Chang , Chih-Yuan Lin , Chun-Fu Lin, An Adaptive Scoring Job Scheduling algorithm for grid computing, Information Sciences: an International Journal, 207, p.79-89, November, 2012.

[11]Robert Elsässer, Burkhard Monien, and Robert Preis. 2000. Diffusive load balancing schemes on heterogeneous networks. In *Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures* (SPAA '00). ACM, New York, NY, USA, 30-38. DOI: https://doi.org/10.1145/341800.341805