Hierarchical Status Information Exchange Scheduling and Load Balancing For Computational Grid Environments

Malarvizhi Nandagopal[1] **and** **Rhymend V Uthariaraj**[2]

[1]Research Scholar, Ramanujan Computing Centre, Anna University Chennai
[2]Professor & Director, Ramanujan Computing Centre, Anna University Chennai

Abstract
The computational grid is a new parallel and distributed computing paradigm that provides resources for large scientific computing applications. It typically consists of heterogeneous resources such as clusters that may reside in different administrative domains, be subject to different access policies and be connected by networks with widely varying performance characteristics. Many researchers have been proposed numerous scheduling and load balancing techniques for locally distributed multiprocessor systems. However, they suffer from significant deficiencies when extended to a grid environment. Computational grids have the potential for solving large-scale scientific computing applications. The main techniques that are most suitable to cope with the dynamic nature of the grid are the effective utilization of grid resources and the distribution of application load among multiple resources in a grid environment. This paper addresses the problem of scheduling and load balancing in a grid architecture where computational resources are dispersed in different administrative domains or clusters which are connected to the grid scheduler by means of heterogeneous communication bandwidths is considered. The proposed work addresses the problem of load balancing using Min-Load and Min-Cost policies while scheduling jobs to the resources in multi-cluster environment. Also, a heuristic taking both the resource load and the network cost into consideration is developed to evaluate the benefits of scheduling jobs to resources in different clusters. In this paper three steps strategy has been used to determine a resource for an arriving job. It also determines the distribution of job to the remote clusters for optimizing the performance. A set of simulations conducted on the GridSim Toolkit showed that the proposed strategy provides significant performance improvement over existing ones.

Keywords:
Grid Computing, Scheduling, Load Balancing, Response Time, Communication Cost.

1. Introduction

Grid computing has emerged as the next-generation parallel and distributed computing methodology that aggregates dispersed heterogeneous resources for solving various kinds of large-scale parallel applications in science, engineering and commerce [1]. In large-scale grid environments, the underlying network connecting them is heterogeneous and bandwidth across resources varies from link to link. Grid environment is extremely unpredictable: processor capacities are different and usually unknown, computers may connect and disconnect at any time, and their speeds may change over time [2].Although load-balancing problem in conventional distributed systems has been intensively studied, new challenges in grid computing still make it an interesting topic and many research projects are under way. Load Balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the grid architectures. Grids have a lot of specific characteristics, like heterogeneity, autonomy and dynamicity, which remain obstacles for applications to harness conventional load balancing algorithms directly. A computational grid is the cooperation of distributed computer systems where user jobs can be executed on either local or remote resources. With its multitude of heterogeneous resources, a proper scheduling and efficient load balancing across the grid is required for improving the performance of the system.

While balancing the load, certain types of information such as number of jobs waiting in queue, load (number of jobs queued on the resource /speed of the resource), job arrival rate, CPU processing rate etc. need to be exchanged among computational resources. A load balancing strategy is categorized as either centralized or distributed [3]. In centralized scheme, all these information need to be stored at one location where load balancing decisions are made. Such centralized schemes also require synchronization among resources. In contrast, in distributed schemes, every node periodically broadcasts its state information throughout the system and executes balancing. However, it requires global information and has the problem of communication overheads incurred by frequent information exchange between resources. Alternatively [4] proposed partitioning nodes into different groups or domains. Each partition has a central or master node to collect the information of all its domain nodes. The information exchanges are between central and its domain nodes only. The proposed work belongs to this category. Load balancing algorithms can be static [5][6] or dynamic [7]. In a static algorithm, the scheduling is carried out according to a predetermined approach. On the other hand, a dynamic algorithm adapts its decision to the current

Manuscript received February 5, 2010
Manuscript revised February 20, 2010

state of the system. Thus, a dynamic approach can be made adaptive to changes in system parameters such as job arrival rate, CPU processing rate, load and communication bandwidth between resources.

In [8], information exchange methods are classified into three categories: periodic, on-demand, and event-driven. In the periodic category, each domain node periodically broadcasts its state information to its central node. In the on-demand category, a node sends a request message to the rest of the nodes when it needs the information. The event-driven category is taking place when some specific conditions are met (events).The proposed work belongs to the periodic category. In [9] and [10], a node collects status information about neighbouring nodes by communicating with them at every load balancing instance. However, for large-scale grid environments, status exchange at each load balancing instance can lead to large communication overhead. Alternatively, estimation technique can be used based on system state information received at sufficiently large interval of time. In ELISA [11], load balancing is carried out based on queue lengths. Whenever there is difference in queue length, jobs will be migrated to lightly loaded processor ignoring job migration cost. This cost becomes important factor when communication latency could be very large such as for grid environment.

Load balancing involves assigning job to a resource proportional to its performance, thereby minimizing the response time of a job. However, there are wide varieties of issues that need to be considered for a heterogeneous grid environment. For example, processing capacities of the resources may differ and their usable capacities may vary according to the load imposed upon them. Further, in grid computing, as resources are distributed in multiple domains in the Internet, not only the computational nodes but also the underlying network connecting them are heterogeneous. Therefore, in the grid environment it is essential to consider the impact of various dynamic characteristics on the design and analysis of scheduling and load balancing algorithms. Due to uneven job arrival patterns and unequal computing capacities, one resource may be overloaded while others may be underutilized. It is therefore desirable to dispatch jobs to idle or lightly loaded resources to achieve better resource utilization and reduce the mean job response time. The strategy proposed here is to perform scheduling and balancing the application load in the grid environment by taking resource heterogeneity, communication delay and network heterogeneity into consideration.

1.1. Literature Review

Previous relevant work includes scheduling in distributed systems [12], [13] and multi-site scheduling [14], where meta-scheduler's decisions are based on predicted load values via time-series analysis. Reference [15] explains a scheduling algorithm on computational grid environment in which the grid Scheduler selects computational resources based on job requirements, job characteristics and information provided by the resources. The main aim of these schedulers is to minimize the Total Time to Release (TTR) for the individual application. TTR includes processing time of the program, waiting time in the queue, transfer of input and output data to and from the resource. The papers [16] and [17] considered static load balancing in a system with servers and computers where servers balance load among all computers in a round robin fashion. It requires each server to have information on status of all computers as well as the load allocated by all other servers. The hierarchical load balancing in grid is referred in [18] where load balancing algorithms are implemented at various levels of grid resources to reduce average response time of the grid application but does not consider the communication cost between clusters.

Most application-level load balancing approaches are based on application partitioning via graph algorithms [20]. However, it does not address the issue of reducing job migration cost, i.e., the cost due to load redistribution. The load balancing algorithms in [19] and [20] proposes a migration of job to balance the load when the nodes become an overloading node but does not consider the resource and network heterogeneity.

When compared with the existing work, the main characteristics of the proposed strategy can be summarized as follows:

- It privileges local load balancing to reduce communication costs.
- Jobs are computation intensive.
- The resources have different processing capabilities.
- Jobs are non pre-emptable which means that their execution on a resource cannot be suspended until completion.
- Jobs are independent which means that there is no communication between them.
- The network bandwidth varies between different cluster resources.

The rest of this paper is organized as follows: Section 2 describes the proposed load balancing model architecture. Section 3 presents the load balancing algorithms at different levels of hierarchy. Section 4 discusses the experimental environment and simulation setup. Section 5 gives the simulation results and finally Section 6 presents the concluding remarks and future work.

2. System Model

A simulation model is used to study the performance of scheduling and load balancing policies in the multi-cluster environment such as grid. In the proposed model job scheduling and load balancing is applied at two levels: grid and local. At grid level, a grid scheduler selects the appropriate cluster for job redistribution based on the proposed Min_Cost policy. At local level local schedulers allocate jobs to computational resources based on the proposed Min_Load policy. In order to explain the proposed model, the topological structure of grid computing is defined.

2.1 Grid Topology

As topological point of view, grid G is considered as a collection of C number of clusters. There is a Global Scheduler (GS) who communicates with each one of the distributed clusters. Each cluster consists of K number of Processing Elements (PEs) and a Local Scheduler (LS). LS acts as a Cluster Manager (CM) and GS act as a Grid Manager (GM). There is a job arrival stream at the LS. LS dispatches the submitted jobs to PE according to the proposed policy. Every cluster is connected to the global network or WAN by a switch. Resources within the cluster are interconnected together by a LAN. An example of such topology is shown in Figure1.

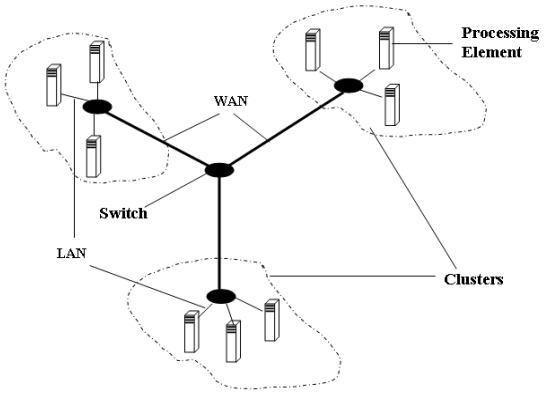

Fig. 1 General Grid Topology

2.2 System Architecture

Figure 2 shows the proposed computational grid architecture. LS handles intra cluster communication and GS handles inter cluster communication. PEs only communicates with the LS in their cluster and do not directly contact resources outside the local cluster for load balancing purposes.

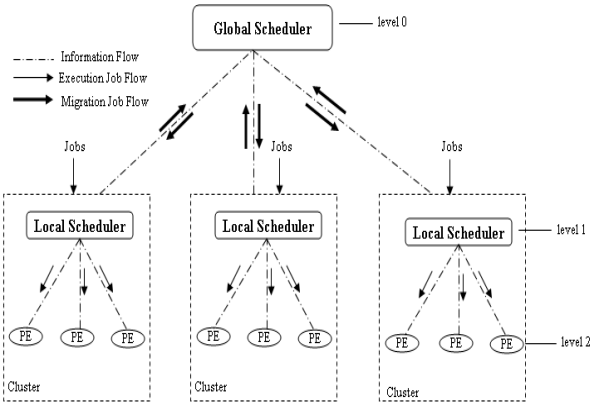

Fig. 2 Structure of System Architecture

The following is a bottom up view of the proposed architecture:

PE Level (level 2): Any workstation called computing unit, can join the grid system and offer its computing resources (PEs) to the grid. When the computing unit starts, it will report information about its computational resources such as CPU speed, CPU load and Processing Capacity to its associated LS.

LS Level (level 1): Every newly joining computing unit registers itself within LS which is responsible for managing a dynamic pool of PEs. The role of LS is to maintain information about active PEs in the same cluster. It also performs load balancing among PEs within its control based on the proposed policy.

GS Level (level 0): The role of GS is to maintain information about active clusters in the grid. In addition to that workload information, GS also maintains the heterogeneous bandwidth details between grid and cluster. Using these informations, GS performs load redistribution among clusters within its control based on the proposed policy.

In the proposed model, jobs are submitted in the local cluster to reduce communication cost induced by job transfer. The submitted jobs are scheduled by LS. Based on the current load of local cluster, LS either assigns job to PEs within its control according to Min-Load policy or transfer the job to GS (saturation case). After getting job from LS, GS redistribute the job to remote cluster resources according to Min-Cost policy. Based on the computed decision the job starts execution. Once started the job run to completion on the assigned resource and are not rescheduled.

3. Scheduling and Load Balancing

The proposed model takes into account the processing capacity, load and bandwidth of the resources in the grid. The class of problem address here is: computation-

intensive and totally independent jobs with no communication between them.

3.1 System Parameters

For each resource participating in the grid the following parameters are defined which will be used later on the load balancing operation.

1) Job Parameters : ID of job, number of instructions per job, job size
2) PEs parameters: CPU speed, workload index which can be calculated using the total number of jobs queued on a given PE and its speed.
3) PE Processing Capacity (PEPC): Number of jobs per second a PE can process. This can be calculated using the CPU speed and an average number of instructions per job.
4) CM Processing Capacity (CMPC): Number of jobs per second the cluster can process. This can be calculated as the sum of the PEPCs of all the processing elements of that cluster.
5) GM Processing Capacity (GMPC): Number of jobs per second that can be processed under the responsibility of GM. This can be calculated as the sum of all the CMPCs managed by the GM.
6) Network Parameter : Bandwidth size
7) Performance Parameters: The following performance parameters are focused: job mean response time, Slowdown, Load Information Traffic and Resource utilization.

3.2 Multi-Level Load Balancing Description

The proposed load balancing model uses a distributed multi-level strategy. The underlined strategy at each level of grid architecture is described as follows:

A) Cluster Level Load Balancing: Consider one LS and its pool of resources (PEs). In the proposed load balancing model only the CPU resources, their processing capacity (PEPC) and load are considered. The total processing capacity of this cluster is given by CMPC. Let NC as the number of jobs arrived at LS at steady-state, the number of jobs to be allocated to each PE will be proportional to the processing capacity of PE in order to maximize the throughput and have a good utilization of all the PEs in the pool. Define the PEShare of each PE in the pool by:

$$PEShare_i = NC.\frac{PEPC}{CMPC} \quad (1)$$

It is also the responsibility of LS to check whether its resources are overloaded by comparing workload index of PE and its PEShare. If the workload index is higher than its share, then LS triggers job transfer policy so that jobs from overloaded PE are transferred to underloaded neighboring PE. At this level of load balancing the job transfer cost is neglected since PEs within the cluster are interconnected by LAN with similar bandwidth.

B) Grid Level Load Balancing: Consider one GS which is responsible for a group of LS. GS maintains information about LS in terms of processing capacity, workload and bandwidth. This is similar to the cluster level load balancing with one additional parameter considered which is job transfer cost. The total processing capacity managed by GS is given by GMPC which is the sum of all the CMPCs. If NG as the number of jobs arrived at GS for redistribution, the number of jobs to be allocated to each CM is also proportional to the total processing capacity of CM. Define the share of each CM is given by:

$$CMShare_i = NG.\frac{CMPC}{GMPC} \quad (2)$$

It is also the responsibility of GS to check whether its resources are overloaded by comparing workload index of CM and its CMShare. If the load index is higher than its share, then GS triggers job migration policy so that jobs from overloaded clusters are migrated to underloaded neighboring clusters. The chosen underloaded clusters are one which needs minimal job transfer cost by adding a network cost heuristic with load metric.

The main advantage of this model is to provide privilege to local load balancing first (within cluster) and then on the entire grid (between clusters). In the proposed design if any resource joins or leaves the grid system its status information is collected and maintained by its higher level scheduler. All these resource status informations are used for the load balancing operation.

3.3 System Design

The proposed grid system consists of M heterogeneous resources, (hereinafter the terms resource, CM and GM are interchangeably used) R_1, R_2, ... R_M, connected via communication network. Each resource has an infinite capacity buffer to store jobs waiting for execution. In Computational Grid environment, as resources are geographically distributed at different locations, the job transfer time from one location to another is a very significant factor for load balancing. Further, the communication latency is very large for WAN through which grid resources are normally connected. Moreover, due to network heterogeneity, the network bandwidth varies from one link to another. Due to these reasons, one cannot ignore the job transfer cost when making a job migration decision. Further, when resources are heterogeneous, jobs are assigned to resources according to its performance.

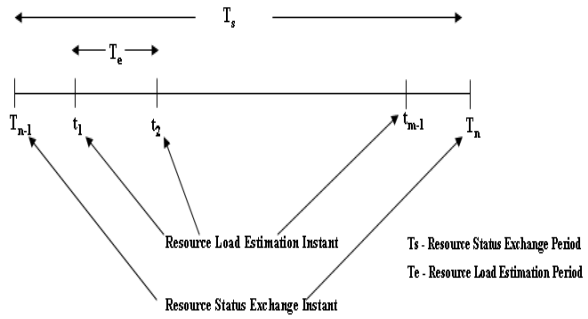

Fig. 3 Resource Status Exchange Intervals

As shown in figure 3, at each periodic interval T_s, each resource in the system calculates its status parameters such as job arrival rate, service rate, and load on the resource and exchanges its status information with its associated manager. The instant at which this information exchange takes place is called a status exchange instant. In the above figure T_{n-1} and T_n represent the resource status exchange instant. As each resource balances the load within the hierarchy, for every resource R_i, in the hierarchy its associated manager calculates the load on R_i's neighboring resource R_k at every estimation instant. In the above figure t1, t_2, ….t_{m-1} represent resource load estimation instants. Based on this calculated load, each manager resource will make a decision of job migration if load in one resource is greater than the load in another resource. It will try to distribute the load based on the load balancing strategy described in the next section.

3.4 Scheduling and Load Balancing Strategy

At any load balancing level the following three steps strategies are used. As the description is in generic way, the concept of group (G) and element (E) is used. Depending on cases a group designs either a cluster or the grid (level 1 or level 0) and an element is a group component (PE of level 2 or cluster of level 1). The main steps of the proposed strategy can be summarized as follows:

Step 1: Workload Estimation

1. For every element E_i and at each status exchange instant period T do
   Send its workload LOD_i to its associated scheduler.
End For
2. At each period T the group node does the following:
   a) Computes its speed and capacity
   b) Evaluates its current workload LOD_G
   c) Calculate element's maximum share.
   d) Send element workload to its associated scheduler.
3. At each estimation instant t, group node calculates the load of neighboring element for each element under its control.

Step 2: Decision Making

For each element do the following

a) Compare the element load LOD_i with the maximum amount of share of that element $EShare_i$. The value of $EShare_i$ depends on the number of jobs arrived at G and the processing capacity of E_i and G_i
b) If LOD_i is greater than $EShare_i$ then the element is in overloaded state.
c) For an overloaded case, determine the overloaded elements (source) and the underloaded ones (destination) to transfer a job from overloaded elements to underloaded ones.

Step 3: Job Transferring

In order to transfer jobs from overloaded elements to underloaded ones the following heuristics are proposed:

Transfer criteria

Switch
G= Cluster:
Perform Min-Load Policy; Return;
G= Grid:
Perform Min-Cost Policy; Return;
End Switch

Min-Load Policy:

This policy is implemented in LS which determines a method a PE is selected for a job submitted in the local cluster. According to this policy, based on collected resource status information LS monitors the load of each PE and selects the PE with least load. In case there are two PEs with identical load, any one is selected at random.

Min-Cost Policy:

This policy is implemented in GS which determines the way a remote cluster is selected for a job migrated from the local saturated cluster. GS calculates the minimum communication cost of sending jobs to remote cluster resources based on the information collected in the last exchange interval. GS selects the cluster that provides minimum overall cost.

Job Selection Criteria

FCFS: Transfer the first submitted job (oldest job). FCFS scheduling policy is applied for jobs waiting in queues, both at GS and LS. FCFS ensures certain kind of fairness, does not require an advance information about job execution time, do not require much computational effort, and is easy to implement.

3.5 Communication Cost

Since resources within a cluster usually use the same file system and there are dedicated file servers, program and data file do not have to be transferred when a job is scheduled to run on a resource in the same cluster (local jobs). Hence the network cost of remote job execution can

be ignored in the single cluster environment. However, in the multi-cluster environment the related files of a job need to be transferred through much slower Internet links if the job is scheduled to run in a remote cluster. Therefore the cost of file transfers must be taken into consideration in the scheduling and load balancing strategy. Experimental results have shown that even though the network cost may be small compared to the job execution time, mean response time and utilization can be substantially improved if network cost is taken into consideration.

Consider the distribution of job J submitted in grid node G from cluster E which is scheduled to run on a resource in a cluster C. The completion time of job *J* is approximated by the following formula:

$$Runtime(J) \cdot \frac{Runqueuelength(C)}{Speed(C)} + \frac{Jobsize(J)}{Linkspeed(G,C)} \quad (3)$$

where the meanings of the variables are listed in Table 1.

Table 1: The Meaning of Variables in Formula (3)

Runtime(J)	Execution time of job J when it runs by itself on a computer with relative speed 1
Jobsize(J)	Total size of job J
Linkspeed(G,C)	Speed of communication link between G and C in Mbps
Runqueuelength(C)	Current run queue length of cluster C
Speed(C)	Speed of cluster C

The term $Runtime(J) \cdot \frac{Runqueuelength(C)}{Speed(C)}$ in (3) estimates the elapsed time of job *J* on cluster C and the term $\frac{Jobsize(J)}{Linkspeed(G,C)}$ in (3) estimates the job transfer time.

The job should be scheduled to the cluster with the minimum expected completion time. Unfortunately, the run time of a job is usually not known when the job arrives. Even if the job size is known, it does not help much in making the scheduling decision without knowledge of the job run time. Notice that Runtime(J) does not depend on the speed of the cluster (see Table 1 for definition). Therefore, formula (3) can be divided by Runtime(J) without affecting the scheduling decision:

$$\frac{Runqueuelength(C)}{Speed(C)} + \frac{Jobsize(J)}{Runtime(J)} \cdot \frac{1}{Linkspeed(G,C)} \quad (4)$$

The value computed by the above formula is referred to as the relative completion time of the job. $\frac{Jobsize(J)}{Runtime(J)}$ is the only unknown term in (4) and use a constant *CT* to replace it. Thus, formula (4) becomes:

$$\frac{Runqueuelength(C)}{Speed(C)} + CT. \frac{1}{Linkspeed(G,C)} \quad (5)$$

This is a heuristic load metric that takes link speed into consideration. It can be used to evaluate the benefits of sending jobs to cluster resources. The cluster that provides the smallest value according to formula (5) should be selected. The constant *CT* is selected such that *CT*. $\frac{1}{Linkspeed(G,C)}$ is much smaller than $\frac{Runqueuelength(C)}{Speed(C)}$. This is because, by adding the network cost term *CT*. $\frac{1}{Linkspeed(G,C)}$ in the load metric, a job has higher probability to be scheduled to clusters that introduce less (or no) network costs. Therefore, for short jobs where network cost is comparable to run time, the completion time using this load metric would decrease significantly. On the other hand, for long jobs where network cost is negligible compared to run time, the completion time would not be affected very much. As a result, the overall system performance, especially mean response time is improved.

Notice that since no network costs are incurred if a job is submitted to run in the local cluster, the second item in formula (4) or (5) (i.e., the file transfer costs) is set to 0 for local schedulers.

3.6 Performance Metrics

The following metrics were selected to evaluate the performance of the proposed model:

1. Mean Response Time: Response time rj of job j is the time period from the job arrival to the completion time of the job. i.e., the time spent in the resource queue plus the job service (execution) time. The mean response time RT:

$$RT = \frac{1}{N} \sum_{j=1}^{N} r_j \quad (6)$$

where N is the total number of processed jobs.

2. Slowdown: Slowdown S_j of a job j is the job's response time divided by the job's execution time. If ej is the execution time of a job j, then the slowdown is defined as follows:

$$S_j = \frac{r_j}{e_j} \quad (7)$$

The average slowdown SLD is

$$SLD = \frac{1}{N} \sum_{j=1}^{N} S_j \quad (8)$$

where N is the total number of processed jobs.

3. Load Information Traffic (LIT): It is a metric for estimating the extent of traffic from clusters to GS due to load information. Let w be the message weight from each cluster and e be the number of load information exchange events occurred until the end of simulation. LIT is defined as follows:

$$\text{LIT} = \sum_{1}^{e} (C.w) \qquad (9)$$

where C is the number of clusters.

4. Experimental Environment

4.1 GridSim Simulation ToolKit

The simulation was carried out on the excellent grid simulation toolkit GridSim ToolKit 4.0 [21] which allows modeling and simulation of entities in grid computing systems-users, applications, resources, and resource load balancers for design and evaluation of load balancing algorithms. A heterogeneous grid environment by using various resource specifications was built. It proposes the method of creating a user job and different types of heterogeneous resources. The resources differ in their operating system type, CPU speed, RAM memory, MIPS rating. In GridSim, application jobs are modeled as Gridlet objects that contains all information related to the job and the execution management. Details of the available Grid resources are obtained from Grid Information Service (GIS) entity that keeps track of the resources available in the grid environment. The experimental environment consisting of hierarchy of resources used for the evaluation of proposed algorithm is shown in figure 4. A grid resource (GS) maintains information about machines (LS) and each machine contains PEs running at different speeds.

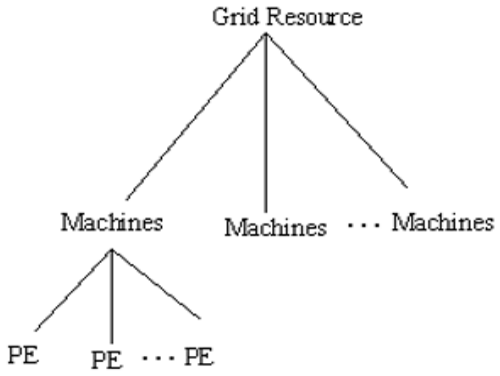

Fig. 4 GridSim Resource Hierarchy

4.2 Simulation Setup

All simulations are performed on a PC (Core 2 Processor, 3.20GHz, 1GB RAM) and all of the time in this paper is the simulation time. The bandwidth speed of low capacity link (within machine) is 10Mbps and the high capacity link (between machines) varies from 0.5Mbps to 100Mbps. All time units are in seconds so the performance metrics are also measured in seconds.

5. Simulation Results and Analysis

The simulation results presented describe the performance of the proposed policies. The proposed policies are compared with Random_GS and Random_LS Policies described in [22]. According to these random policies a resource for job execution is selected randomly without considering its load and cost needed to transfer a job to that resource. However, in the proposed policy the resource for job execution is selected by considering load metric along with network cost which improves mean response time significantly. After 500 jobs, Min_Load and Min_Cost policies took the time 39.8% less than the Random_LS and Random_GS policies did. From Fig 5 and Table 2 it is observed that the proposed policies can increase performance by reducing the mean response time as compared to no load balancing and random polices case.

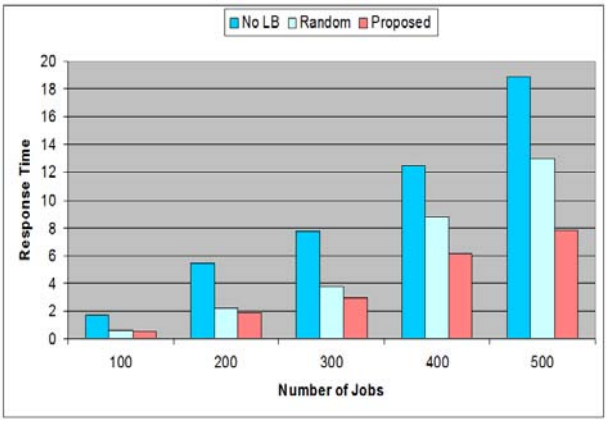

Fig. 5 RT versus number of jobs

Table 2: RT Analysis

No. of Jobs	No Load Balancing	Random Policies	Proposed Policies	Improvement (%)	Unit Improvement(%)
100	1.667	0.556	0.512	7.913	0.0791
200	5.487	2.188	1.875	14.305	0.0715
300	7.767	3.772	2.916	22.693	0.0756
400	12.452	8.789	6.152	30.003	0.0758
500	18.892	12.978	7.812	39.805	0.0796

Figure 6 illustrates the relative decrease in SLD (DSLD) when the proposed Min-Cost and Min-Load policies are employed instead of Random_GS and Random_LS. The proposed policies yield the highest DSLD for all jobs.

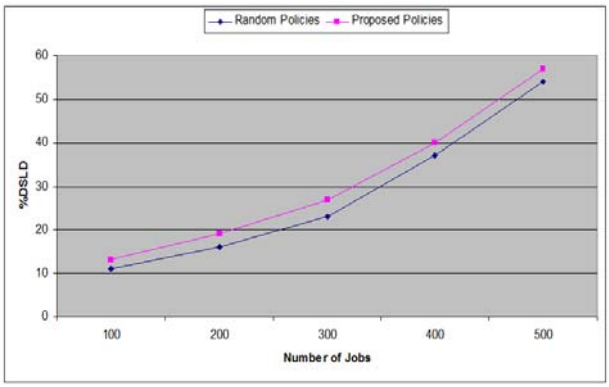

Fig. 6 Performance analysis of DSLD %

The policies described in Section 3 are characterized by the resource status interval (T_s) parameter. The value of this parameter significantly affects LIT and RT. A high T_s value increases the number of jobs waiting in scheduler's queue and their delay due to scheduling deferment and thereby RT increases. On the other hand, a small T_s value eliminates this problem but increases the overhead as resource load information is required more frequently and therefore increases LIT. Figure 7 shows how RT and LIT is affected with regard to T_s.

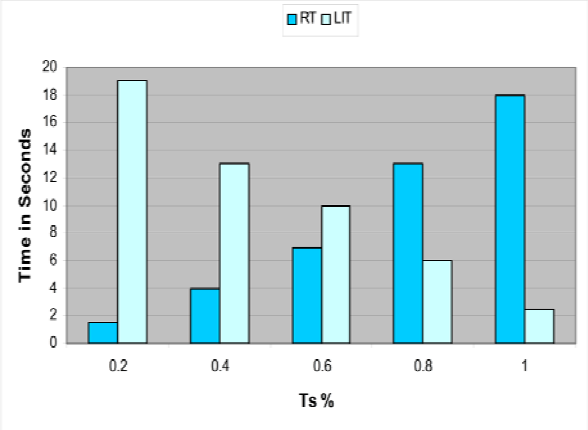

Fig. 7 Variation of T_s

6. Conclusion and Future Work

This paper addressed the problem of scheduling and load balancing for computational grid environment. Load balancing strategies in the multi-cluster environment is proposed where clusters are located in different local area networks which are physically wide apart from one another. The proposed load balancing model takes into account the heterogeneity of the computational and network resources. i.e., the resources are with different processing capacity and network bandwidth. The load balancing policies at various levels of hierarchy are proposed to optimize various performance metrics. In addition, a heuristic that considers both machine load and network speed is suggested to estimate the completion time of executing jobs in remote clusters. Performance results have indicated that the proposed approach improves system performance in terms of mean response time and average slowdown.

Now we discuss some of the limitations of this work and present some possible directions for future research. In this work, we assume that there is no precedence constraint among different jobs or different tasks of a job. Usually, the jobs are independent of each other in the grid, but different tasks of a job may have some precedence constraints. Hence, it is an interesting direction for future research. Such dependencies will not only make the problem extremely difficult to solve, but would also require estimating a very large number of parameters. In future we should also consider some fault tolerant measures to increase the reliability of our algorithm.

References

[1] Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a Future Computing Infrastructure, 2nd Edition. Morgan Kaufmann, San Mateo (2004).

[2] Dobber, M., Koole, G., Mei, R.: Dynamic load balancing experiments in a Grid. In: Proceedings of IEEE International Symposium on Cluster Computing and the Grid, Cardiff, 2005.

[3] Kai Lu, Riky Subrata and Albert Y. Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites", IEEE International Performance, Computing, and Communications Conference, 2006. IPCCC 2006. Vol 25Page(s):9 pp. – 320.

[4] Kimura, K., Ichiyosi, N.: Probabilistic analysis of the optimal efficiency of the multi-level dynamic load balancing schemes. In: Proceedings of the 6th Distributed Memory Computing Conference, Portland, (1991)

[5] Kameda, H., Li, J., Kim, C., Zhang, Y.: Optimal Load Balancing in Distributed Computer Systems. Springer, London (1997).

[6] H. Attiya, "Two Phase Algorithm for Load Balancing in Heterogeneous Distributed Systems," Proc. 12th Euromicro Conf.Parallel, Distributed and Network-Based Processing (PDP '04), p. 434,2004.

[7] HAN Xiangchun, PAN Xun. Distributed scheduling pattern for dynamic load balance in computing grid [J]. Computer engineering and Design,2007,28(12):2845-2847.

[8] Shah, R., Veeravalli, B., Misra, M.: On the design of adaptive and de-centralized load balancing algorithms with load estimation for computational Grid environments. IEEE Trans. Parallel Distrib. Syst. **18**, 1675–1686 (2007).

[9] Oliker, L., Biswas, R., Shan, H., Smith, W.: Job Scheduling in Heterogeneous Grid Environment. LBNL-54906. Lawrence Berkeley National Laboratory,Berkeley (2004)

[10] Shan, H., Oliker, L., Biswas, R.: Job superscheduler architecture and performance in omputational Grid environments. In: Proceedings of IEEE/ACM Conference on Supercomputing, Phoenix, (2003).
[11] Anand, L., Ghose, D., Mani, V.: ELISA: an estimated load information scheduling algorithm for distributed computing systems. Int. J. Comput. Math. Appl. **37**(8), 57–85 (1999)
[12] H.D. Karatza, "Performance of Gang Scheduling Policies in the Presence of Critical Sporadic Jobs in Distributed Systems", Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems-SPECTS 2007, San Diego, CA, 2007, pp. 547-554.
[13] H.D. Karatza, "Scheduling Gangs in a Distributed System", *International Journal of Simulation: Systems, Science Technology*, UK Simulation Society, Vol. 7, no. 1, (January): 15-22, 2006.
[14] M. Ioannidou, H. Karatza, "Multi-site Scheduling with Multiple Job Reservations and Forecasting Methods", In Proceedings of International Symposium on Parallel and Distributed Processing and Applications, volume 4330 of Lecture Notes in Computer Science, Springer, 2006, pp. 894-903.
[15] N.Malarvizhi,V.Rhymend Uthariaraj , "A New Mechanism for Job Scheduling in Computational Grid Network Environments " in proceedings of 5th International Conference on Active Media Technology ,Volume 5820 of Lecture Notes in Computer Science, Springer, 2009,pp. 490-500.
[16] Grosu, D., Chronopoulos, A.T.: Noncooperative load balancing in distributed systems. J. Parallel Distrib. Comput. 65(9), 1022–1034 (2005)
[17] Penmatsa, S., Chronopoulos, A.T.: Job allocation schemes in computational Grids based on cost optimization. In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium, Denver, (2005).
[18] N.Malarvizhi,V.Rhymend Uthariaraj, " Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment" in *Proc. Int. Conf on Advanced Computing, India, Dec* 2009, pp. 97-104.
[19] H. Johansson and J. Steensland, "A performance characterization of load balancing algorithms for parallel SAMR applications," Uppsala University, Department of Information Technology, Tech. Rep. 2006- 047, 2006.
[20] Y. Hu, R. Blake, and D. Emerson, "An optimal migration algorithm for dynamic load balancing," *Concurrency: Practice and Experienc*e, vol. 10, pp. 467–483, 1998.
[21] Buyya R, "A Grid simulation toolkit for resource modeling and application scheduling for parallel and distributed computing". www.buyya.com/gridsim/.
[22] Zikos, S., Karatza, H.D., 2008. Resource allocation strategies in a 2-level hierarchical grid system. In: Proceedings of the 41st Annual Simulation Symposium (ANSS), April 13–16, 2008. IEEE Computer Society Press, SCS, pp. 157–164.

Malarvizhi Nandagopal received her BE and ME degree in computer science and Engineering from Madurai Kamaraj University . She is currently a Research Scholar in Ramanujan Computing Centre and is pursuing her PhD Degree in Anna University Chennai.Her research interest includes Parallel and Distributed Computing, Cloud Computing and Grid Computing. She is a member of IEEE and WIE.

V.Rhymend Uthaiaraj received his PhD degree in computer science and Engineering from Anna University Chennai. He is currently working as a Professor and Director in the Department of Ramanujan Computing Centre, Anna University Chennai. His area of interest includes Computer Networks, Network Security, Computer Algorithms and Modeling, Mathematical Programming.