Modified Hierarchical Load Balancing Algorithm for Scheduling in Grid Computing

Heena Setia
PG Student
Department of Computer Science & Engineering
Giani Zail Singh Campus College of Engineering & Technology Bathinda, India

Abhilasha Jain
Assistant Professor
Department of Computer Science & Engineering
Giani Zail Singh Campus College of Engineering & Technology Bathinda, India

Abstract

Grid computing provides a large reliable framework for execution of large scale applications. Grid is a collection of huge number of resources which are heterogeneous in nature. And it also provides access to those resources for reliable execution of the problem. Computation and data grid are two types of grid. This paper considers computational grid. In computational grid, Scheduling of resources is the main challenge. Resources are of different types and dynamic in nature, so load on resource changes randomly. The previous scheduling optimization algorithms consider either economic factor or time factor along with load balancing. This paper proposed a system that considers both economic and time factor with load balancing and provides a reliable framework that is client efficient algorithm. Experimental results show that modified hierarchical load balancing algorithm (mHLBA) provides a better framework for allocation of resources.

Keywords: Cost, Grid Computing, Load Balancing, Makespan, Scheduling

I. INTRODUCTION

Grid computing made up of two words: 'Grid' and 'Computing'. Grid means a network or matrix and Computing means any computation. Grid computing means a framework in which number of devices compute together for solving high level computation problem. It provides a distributed programming framework in which no. of devices work together on large scale applications[6][8]. Grid computing scheduling framework is described as:

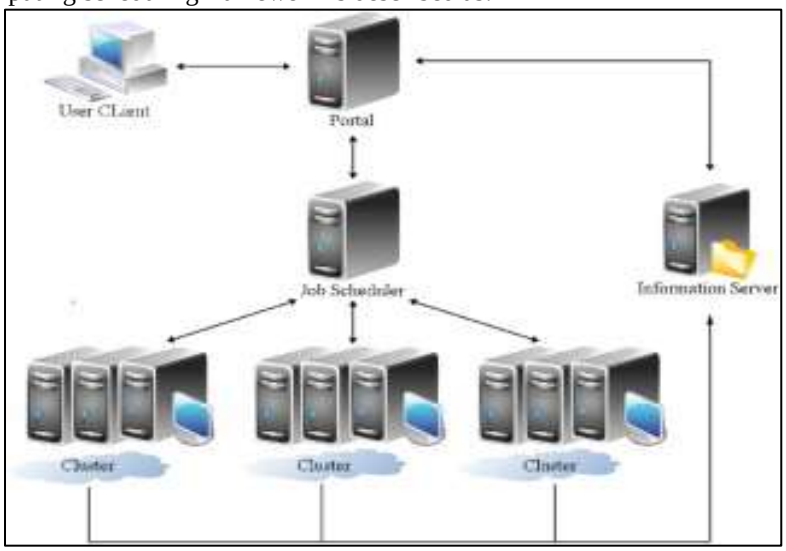

Fig. 1: Scheduling Framework[1]

- Client: End users, who sends their applications to grid scheduler for execution.
- Portal: It is the interface between scheduler and client by which clients submit their jobs.
- Information server: It present as a mediator between portal and clusters. It contains all information about resources (status, computation power, load) & passes it to portal.
- Job scheduler: Scheduler / Broker considers characteristics of both job and resource (received from portal) and making allocation decision.
- Cluster / Resource: Any computing device can be treated as resource in grid computing environment. This proposed algorithm also considers cluster environment.

Scheduling means order of job execution on different resources present in grid infrastructure. Scheduling in grid computing in NP-hard problem[10]. Architecture of scheduler is responsible for the allocation process in grid computing. In meta computers, there are various machines present in grid with their local scheduling policies. So, scheduling on that machines is a complex task. Scheduling may be centralized or decentralized[2]. In centralized scheduling, all information regarding resources is collected at central point. Better scheduling decisions are made, because all necessary information is present at a single point. Centralized scheduling is not scalable. In decentralized scheduling, information regarding resources is distributed on resources. Every resource has a list of all remote resources and resources interact with each other for making allocation decision. Decentralization scheduling is scalable.

Section 1 described the introduction to grid computing scheduling. Section 2, 3 will describe related work & the proposed algorithm respectively. And section 4 will provide experimental results.

II. RELATED WORK

Two types of grid are: computation grid (grid that requires more processing power/different type of computing resources) & data grid (grid that requires large scale data management)[12][13].

Karatza et.al in [11] Load balancing algorithm can be static (all parameters are predefined) or dynamic (random parameters) algorithms. Load balancing in grid computing can be defines by five polices that are: Information policy, triggering policy, resource type policy, location policy and selection policy.

George et.al in [15] They proposed an LBMM (Load Balanced Min- Min) algorithm as an improvement of Min-Min algorithm. In Min-Min algorithm, task allocated to that resource, which takes minimum time to execute (means completion time is minimum). After this min-min allocation, it identifies the heavy load resource & assign task to idle resource available to balance load and improves makespan. In this rescheduling process, the completion time of tasks is less than the makespan produced by Min-Min algorithm.

Lee et.al in [1] they proposed a hierarchical load balancing algorithm (HLBA) for computational grid environment. Two main goals of this algorithm are to balance load and minimizes the execution time of gridlets. For computational grids , computing power is the main focus. Then, it first calculates computing power of all resources present in grid infrastructure. When job submitted to grid broker, it selects the resource with highest computing power. After that, (for load balancing) average load on selected resource (AL) is calculated and compare with threshold value. If AL is less than the balance threshold, then job is assigned to that cluster and local updates are done. Otherwise, mark that selected cluster overloaded and choose cluster with second highest computing power and repeat procedure until allocation is done.

Keerthika et.al in [3] They proposed an hybrid approach known as multi criteria scheduling algorithm(MCSA). This algorithm is used for computational grids that considers two factors: load balancing and user deadline. In scheduling architecture, GIS have information about all resources (capacity, ID, load factor, speed) and when user submit the task with its length and user deadline to grid broker, broker schedules it to the proper resource. Here, machine is collection of PEs and act as PEM (processing entity manager). Here, it calculates ETC (expected time to compute), completion time of gridlets, total completion time of all gridlets (tasks) by considering different parameters. The main goal of this algorithm is load balancing. So it calculates load of each PE, machine and resource mathematically & load on resource is checked against threshold value calculated. For allocation, average load on resource should be less than threshold value. Same steps are repeated for each set of tasks. In short, it follows proactive fault tolerance, user deadline and load balancing for allocation.

Jain et al in[4] They proposed a fault tolerant algorithm based on the concept "Job Checkpoints". Checkpoints are very helpful because it saves the enough state information of program running and if the program failed at middle, can execute it later from last checkpoint. This algorithm mainly consider number of checkpoint intervals and duration of checkpoint interval where checkpoint interval is the length between two checkpoints. Checkpoints reduce job execution time, but at the cost of huge memory. Checkpoint interval should be of optimal length (neither too shorter nor too longer). In this algorithm, grid simulation toolkit is used to register resources in GIS. They use terms fault index (calculate occurrence of faults), number of checkpoints. Broker allocates jobs to resources according to any random algorithm. Example - FIFO. If job fails during processing, it increment fault index, otherwise decrements. If fault index is a positive number, then number of checkpoints is equal to one plus to fault index. If job fails before checkpoint 1, then start job from beginning otherwise from last checkpoint. Results show that this algorithm improves time of computation.

III. PROPOSED ALGORITHM

A. Problem Definition

Many algorithms are developed for scheduling in grid computing environment that have considered either economy factor or time factor but not both factors together. This paper develops a system that considers the cost and time factor with load balancing (For example: in flight system, one can filter the flights according to time factor/ cost factor/ time + cost factor). Time and cost factors are inversely proportional to each other. This paper develops a client efficient algorithm that considers both factors together to generate an optimal solution. This system will consider all the parameters and will help in solving the existing problem of task scheduling algorithm in the grid based environment.

B. System Framework

This algorithm considers a cluster framework in which cluster/resource is collection of machines and machine is group of processing units. Machine is the manager of all processing units. Clusters in this grid environment can have different no. of machines, but all machines in different clusters having the same no. of processing units. In this, we consider a static framework, in which it defines the no. of clusters, machines and processing units.

In this system architecture, user sends tasks to grid scheduler. GIS(grid information server) having all information regarding status of resources present in environment. Scheduler allocate resource to task according to the algorithm and policy defined.

C. Steps in algorithm

- When user submit jobs to scheduler, then scheduler passes request to information server for fetching all parameters regarding status of available resources in grid environment. This algorithm works on computational grid. So, computing power treated as main characteristics. Here, cluster is a collection of machines. So, CP (computing power) of cluster is calculated by average of CP of all machines in cluster. The formula is given as:

$$\text{CPi} = \sum_{n=1}^{k}(\text{Speed\_of\_CPU}_n * \text{Idle\_CPU\_percentage}_n)/\text{n} \qquad (1)$$

  Where Idle CPU percentage is calculated by subtracting utilized part of CPU in percentage from one $(1 - \text{Utilized\_CPU\_percentage}_\text{n})$. Here, CPi is computing power of cluster i, k is no. of machines in cluster i, Speed_of_CPUn is speed of machine n in MIPS (million instructions per second) which present in cluster i. After it calculated CP of all clusters, it arranged resources in increasing order of their computing power value & select resource with highest value of CP.
- After selection of optimal resource with highest CP, it will proceed for load balancing procedure. For which it will calculate load on cluster. As mentioned above cluster is a collection of machines. So, first it calculate load on each machine. Load on each machine is calculated by weighted sum of squares method given as:

$$\text{Load}_{\text{k,i}} = \sqrt{\sum_{i=1}^{m}(w_i \text{L}_\text{i}^2)} \qquad (2)$$

  Here, $\text{Load}_{\text{k,i}}$ is the load of each machine k in cluster i & m is the total no. of clusters present in grid architecture. w_i is the weighted value & L_i is the load attribute. And $\sum_{i=1}^{m}(w_i) = 1$
- In mHLBA, it considers three load attributes that are CPU utilization, memory utilization and network utilization. So, the formula mentioned in (2) can be written as:

$$\text{Load}_{\text{k,i}} = \sqrt{\alpha \text{C}_\text{k}^2 + \beta \text{M}_\text{k}^2 + \gamma \text{N}_\text{k}^2} \qquad (3)$$

  Here, C_k , M_k, N_k are load attributes for CPU, memory and network utilization respectively. And α, β, γ are the weighted values for CPU, memory and network utilization respectively.
- Load on each cluster is calculated by average of load on all machines present in that cluster. Formula for calculating load on cluster is given as:

$$\text{LC}_\text{i} = \frac{\sum_{n=1}^{m}(\text{Load}_{\text{n,i}})}{\text{m}} \qquad (4)$$

  Here, LC_i is the average load on cluster i, $\text{Load}_{\text{n,i}}$ is load of machine n in cluster i & m is the total no. of machines in cluster i.
- Load on system is now calculated, which is equal to average of load on each cluster presents in grid architecture. Formula is given as:

$$\text{LS} = \frac{\sum_{n=1}^{k}(\text{LC}_\text{n})}{\text{k}} \qquad (5)$$

  Here, LS is the average load on system, LC_n is the load on cluster n & k is the total no. of clusters in grid environment.
- Now it defines a threshold[14] value for differentiate between overloaded and under-loaded resources. If average load on cluster is greater than threshold value, then resource marked as overloaded resource else under-loaded. Threshold value is calculated as:

$$\Psi = \text{LS} + \sigma \qquad (6)$$

  Here, σ is Standard deviation of load on the system and calculated as:

$$\sigma = \sqrt{\frac{\sum_{n=1}^{K}(\text{x} - \bar{\text{x}})^2}{\text{K}}}, \text{ for all n.} \qquad (7)$$

  Here, x is the average load on the system(LS) & $\bar{\text{x}}$ is the average load on the cluster(LC) & K is no. of clusters.
- Cost of task execution depends on the resource cost per MI(million instruction) & Task length in MI. Resource budget is calculated as:

$$\text{RB} = \text{RC} * \text{TL} \qquad (8)$$

  Here, RB is the resource budget, RC is the resource cost per MI & TL is the task length in MI.
- Makespan (Time of computing/ Execution time) is depend on two factors that are: capacity of resource (Computing Power) & length of task in MI. Calculated as:

$$\text{MS} = \frac{\text{TL}}{\text{CP}} \qquad (9)$$

Here, MS is the makespan of gridlet (task).

- Average budget of tasks executed is calculated as:

$$AB = \frac{\sum_{i=1}^{n} RB}{n} \quad (10)$$

Here, AB is the average budget & n is no. of gridlets(tasks) submitted.

- Average makespan of gridlets submitted in one round is calculated as:

$$AM = \frac{\sum_{i=1}^{n} MS}{n} \quad (11)$$

Here, AM is the average makespan & n is no. of tasks submitted in one round.

D. Scheduling Flow

When gridlet arrives, scheduler/broker in grid architecture will select the resource with highest computing power (CP). After that it should check these three conditions:

- Load Balancing: Load on selected resource/cluster (LC) should be less than the threshold value (calculated by using formula in equation 6). This will make sure that the selected resource is not heavily loaded.
- Cost Effectiveness: Cost of computing task should be favorable. For this condition, this algorithm defines an optimal budget (that a person decides for his/her task, thus need to keep it random) for each task at the time of initialization. If resource budget (calculated by using formula in equation 8) of selected resource for execution of task is less than that of optimal budget, it will choose the resource budget otherwise optimal budget. This procedure will provide us resource with favorable cost.
- Makespan: Makespan means execution time of gridlet. After obtaining the value for computation time (calculated by using formula in equation 9), the value of makespan for selected resource should be less than gridlet time value (deadline decided by user when initialized the scheduling process).

If the above three conditions are satisfied, then scheduler will allocate gridlet to selected resource. Else, it will go for resource with second highest computing power & check conditions again. If scheduler does not find any resource with all conditions satisfied, then it will choose resource according to the criteria considered in HLBA.

When job submitted to a resource for execution, local updates are done in resource status (all parameters are recalculated). And when job completed its execution, global updates are done in resource status (resource is available for other task).

The flow of mHLBA is given as:

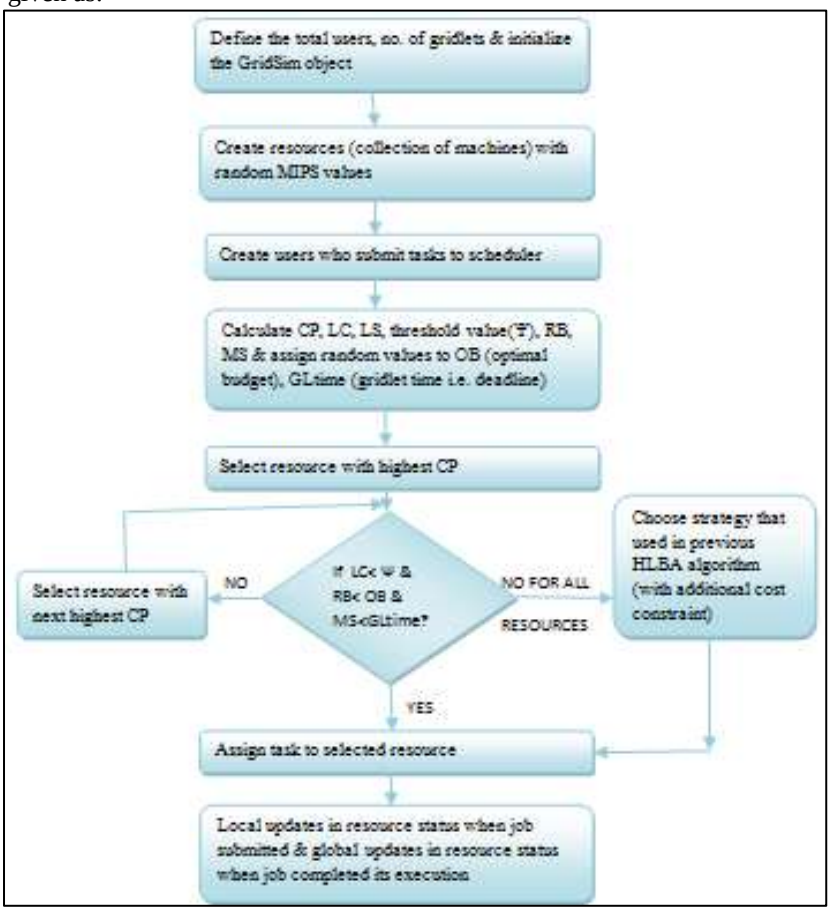

Fig. 2: Flow of mHLBA

IV. SIMULATION & EXPERIMENT RESULTS

A. GridSim Simulator

For modeling and simulation of dynamic resources and tasks to evaluate performance of scheduling algorithms, distributed system framework is required. Building that type of infrastructure is a very expensive process. Also, even if test-infrastructure is available, it is limited to no. of resources and domains, & evaluating scheduler performance for large scale applications is harder. In this algorithm it uses GridSim for creating a virtual environment for evaluating performance of proposed algorithm.

GridSim is a Java based simulation toolkit[5][9]. It provides a facility for simulation of different classes of dynamic resources, users, applications, brokers. It also used to simulate application schedulers for single/multiple administrative domain(s) system such as clusters and grids.

B. Result Analysis

This section is divided into two parts describes below:

1) Based on Weighted Attributes

In this algorithm, it calculates load on resource for optimal allocation. In equation (3), it considers three weighed values for load attributes that are α (CPU utilization), β (Memory utilization), γ (Network utilization). This algorithm can take different values for these attributes (summation of all three attributes is one). Simulation results are shown below by taking three different sets of values for these parameters.

Table – 1
Simulation parameters

Parameters	*Values*
No. of tasks	*2000*
Size of tasks(MI)	*300,000-500,000*
No. of machines in cluster	*4*
CP of resource(MIPS)	*500-5000*
No. of clusters	*3*
Baud Rate	*500-1000*
Resource Budget	*200-500*
No. of tasks in 1 round	*40*
Gridlet Time	*500-1500 sec*

Table – 2
Graph parameters (Makespan v/s Weighted values)

Weighted Values	*Average Execution time in seconds*
α=0.6,β=0.3,γ=0.1	*3612*
α=0.3,β=0.6,γ=0.1	*4764*
α=0.1,β=0.3,γ=0.6	*4950*

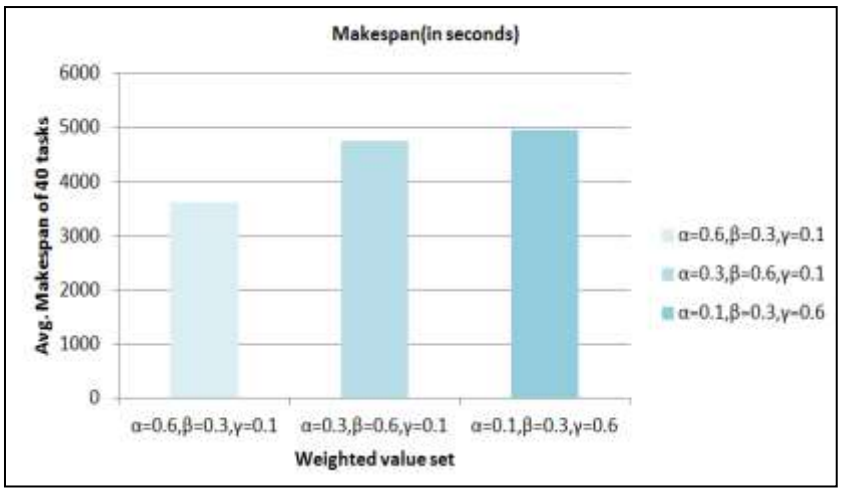

Fig. 3: Makespan for each weighted value set

Above graph shows that more value for CPU utilization, less the execution time. Execution time will increase as this algorithm considers low value for CPU utilization. If network or memory utilization is more, then scheduler either provide less reliable results or allocate task to heavily loaded resource. CPU utilization factor is most important here for fast execution. If computing power is high, then resource executes task fast. So, in this algorithm the optimal value set is α=0.6,β=0.3,γ=0.1.

2) Based on Cost and Time Factor

It considers optimal weighted value set, then it compares average cost and time of computing tasks of (Calculated by using formula in equation 10 & 11) proposed algorithm (mHLBA) with HLBA[1]. Simulation parameters and results are shown below:

Table – 3
Simulation parameters

Parameters	*Values*
No. of tasks	*2000*
Size of tasks(MI)	*300,000-500,000*
No. of machines in cluster	*4*
CP of resource(MIPS)	*500-5000*
No. of clusters	*3*
Baud Rate	*500-1000*
Resource Budget	*200-500*
No. of tasks in 1 round	*40*
Gridlet Time	*500-1500 sec*

Table – 4
Graph parameters (Makespan of hlba &mhlba)

Algorithms	*Average Execution time in seconds*
mHLBA	*3613*
HLBA	*4648*

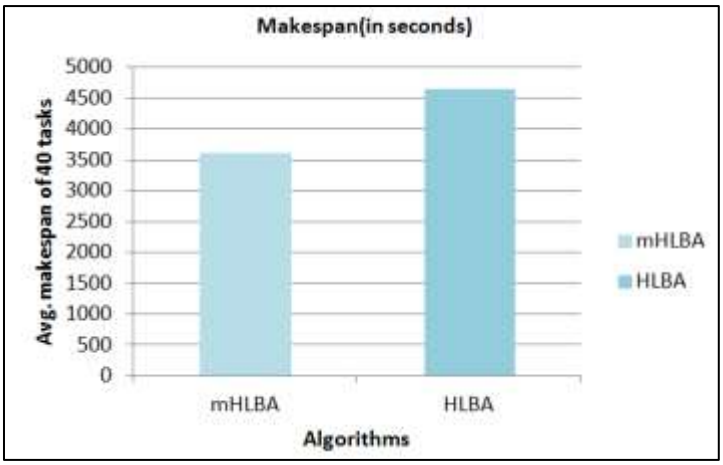

Fig. 4: Average Makespan for HLBA & mHLBA

The above graph shows that execution time of mHLBA is less than HLBA. This algorithm applies conditions to choose optimal resource which takes lesser time to compute. This algorithm also consider budget for computing tasks. Average cost of computing tasks for mHLBA & HLBA is compared as:

Table – 5
Graph parameters (average cost of computation of mhlba & hlba)

Algorithms	*Average cost of computing 40 tasks*
mHLBA	*270*
HLBA	*420*

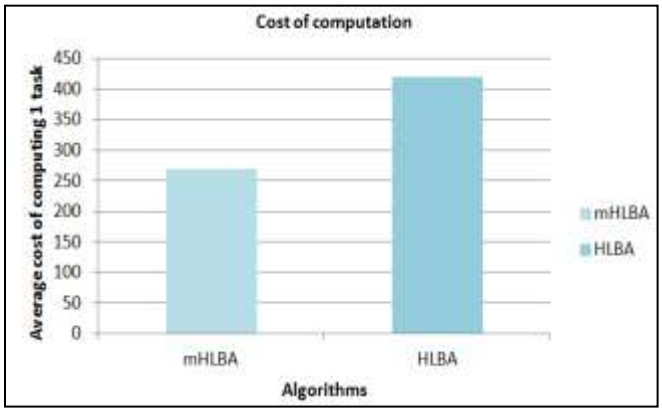

Fig. 5: Average cost of computing of HLBA & mHLBA

The above graph shows that average cost of computation of mHLBA is less than HLBA. Now, it compares total cost & makespan of mHLBA with HLBA by taking same simulation parameters.

Table – 6
Graph parameters (total makespan & cost of computation of mhlba & hlba)

Algorithms	*Total makespan of 2000 tasks*	*Total cost of computing 2000 tasks*
mHLBA	1860637	541900
HLBA	232406	839100

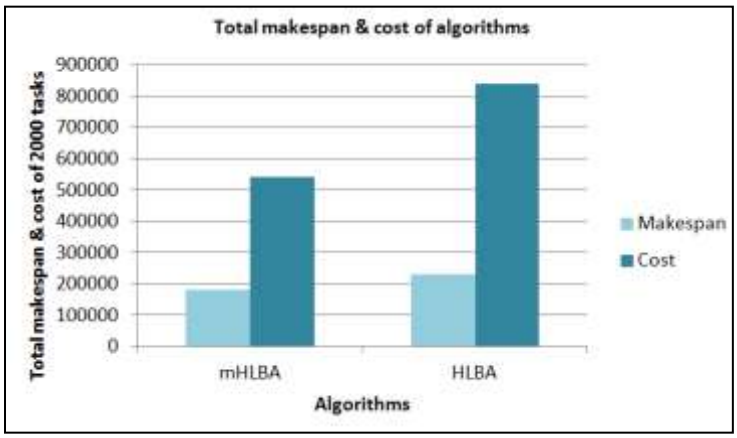

Fig. 6: Total makespan & cost of computing of HLBA & mHLBA

It shows that makespan & cost of mHLBA is less than HLBA. In short it provides an optimal algorithm that satisfies the client by filtering the selection criteria. HLBA does not allocate resources by considering budget & makespan of computing tasks, it considers only load factor for improving makespan. The proposed algorithm (mHLBA) provides an optimal scheduling strategy that considers both factors and then allocates resources.

Results are calculated by average improvement in 10 simulations. On average, mHLBA improves makespan & cost of computing tasks by 22.26% & 35.71% respectively than HLBA.

And it proved in HLBA[1] that performance of HLBA is better than most fit task scheduling algorithm (MFTF) [16] & ACO [7]. Hence, mHLBA is an efficient algorithm with better performance than HLBA, MFTF & ACO.

V. CONCLUSION

Grids are rising as the base for advance computing. The resources in grid computing are distributed, so the performance of grid computing is really hard to manage as sometimes the management of the time and cost constraint is very complex. It also depends on the quality and type of user and the most common requirement of users is to find cost and time efficient algorithm. As time and cost efficient algorithms both separately are not able to provide user oriented results. Because in the present era everyone is looking for the resources with minimal cost and with faster turnaround time or deadline. So, considering both time and cost constraints with load balancing approach is proposed. Experiments have been done for makespan and cost of computing for evaluating performance of the algorithm. This approach improves performance and provides a solution, which is client efficient. Experimental results show that modified hierarchical load balancing algorithm (mHLBA) provides a better framework for allocation of resources in grid computing environment and having performance better than MFTF, ACO & HLBA as specified in result analysis. So, basically when there are multiple resources with the same cost and capability, the mHLBA schedules jobs on them using the time optimization strategy which provides user oriented results. And load balancer plays an Important role to manage the load. So, mHLBA is a really good algorithm and this can be further improved in many ways.

REFERENCES

[1] Lee, Y. H., Leu, S., & Chang, R. S. (2011). Improving job scheduling algorithms in a grid environment. Future generation computer systems, 27(8), 991-998.
[2] Hamscher, V., Schwiegelshohn, U., Streit, A., & Yahyapour, R. (2000). Evaluation of job-scheduling strategies for grid computing. In Grid Computing—GRID 2000 (pp. 191-202). Springer Berlin Heidelberg.
[3] Keerthika, P. and Kasthuri, N., 2013. A hybrid scheduling algorithm with load balancing for computational grid. International Journal of Advanced Science and Technology, 58, pp.13-28.
[4] Jain, S., & Chaudhary, J. (2014, February). New fault tolerant scheduling algorithm implemented using check pointing in grid computing environment. In Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on (pp. 393-396). IEEE.
[5] Buyya, R. and Murshed, M., 2002. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and computation: practice and experience, 14(13-15), pp.1175-1220.
[6] Raicu, I., Foster, I. T., & Zhao, Y. (2008, November). Many-task computing for grids and supercomputers. In Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on (pp. 1-11). IEEE.

[7] Ku-Mahamud, K. R., & Nasir, H. J. A. (2010, May). Ant Colony Algorithm for job scheduling in grid computing. In Mathematical/Analytical Modelling and Computer Simulation (AMS), 2010 Fourth Asia International Conference on(pp. 40-45). IEEE.
[8] Li, F., Qi, D., Zhang, L., Zhang, X., & Zhang, Z. (2006, June). Research on Novel Dynamic Resource Management and Job Scheduling in Grid Computing*. In Computer and Computational Sciences, 2006. IMSCCS'06. First International Multi-Symposiums on (Vol. 1, pp. 709-713). IEEE.
[9] Sulistio, A., Yeo, C. S., & Buyya, R. (2003). Visual modeler for Grid modeling and simulation (GridSim) toolkit. In Computational Science—ICCS 2003 (pp. 1123-1132). Springer Berlin Heidelberg.
[10] A. Di Stefano and G. Morana, " A bio-inspired distributed algorithm to improve scheduling performance of multi-broker grids", Springer Natural Computing, DOI 10.1007/s11047-012-9319-8, vol. 11, no. 4, (2012), pp. 687-700.
[11] H. D. Karatza, "Job scheduling in heterogeneous distributed systems", Journal of Systems and Software, (1994), pp. 203-212.
[12] C. Rosas, A. Sikora, E. Cesar, J. Jorba and A. Moreno, "Improving Performance on Data-intensive Applications Using a Load Balancing Methodology Based on Divisible Load Theory", International Journal of Parallel Programming, DOI: 10.1007/s10766-012-0199-4, (2012).
[13] J. Wu, X. Xu, P. Zhang, C. Liu, "A novel multi-agent reinforcement learning approach for job scheduling in Grid computing". Future Gener, Comput. Syst. 27, pp. 430-439, (2011).
[14] Suri, P. K., & Singh, M. (2010, February). An efficient decentralized load balancing algorithm for grid. In Advance Computing Conference (IACC), 2010 IEEE 2nd International (pp. 10-13). IEEE.
[15] Kokilavani, T., & Amalarethinam, D. D. G. (2011). Load balanced min-min algorithm for static meta-task scheduling in grid computing. International Journal of Computer Applications, 20(2), 43-49.
[16] Wang, S. D., Hsu, I., & Huang, Z. Y. (2005, July). Dynamic scheduling methods for computational grid environments. In Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on (Vol. 1, pp. 22-28). IEEE.