An Optimal Job Scheduling Algorithm in Computational Grids

Ramya R
Student
Department of Computer Science and Engineering
B.S Abdur Rahman University, Chennai.

Shalini Thomas
Assistant Professor
Department of Computer Science and Engineering
B.S Abdur Rahman University, Chennai.

ABSTRACT
Grid computing is an emerging technology that involves coordinating and sharing of resources to carry out complex computational problems. Resource management and scheduling plays a crucial role in achieving high utilization of resources in grid computing environments. Due to heterogeneity of resources, scheduling an application is significantly complicated and challenging task in grid system. Most of the researches in this area are mainly focused on to improve the performance of the grid system. To achieve the performance in grid environment, many Job scheduling algorithms are implemented. Existing approaches of Grid scheduling doesn't give much emphasis on the performance of a Grid scheduler. This paper introduces an algorithm called Optimized Hierarchical Load Balancing Algorithm (OHLBA) for Job scheduling and Load Balancing. The proposed method is to dynamically create an optimal schedule to complete the jobs within minimum makespan. The main contributions are to balance the system load and minimize the makespan of jobs. Our proposed approach uses a Grid simulation toolkit (GridSim) to analyze the performance of OHLBA algorithm with other algorithms in terms of makespan and efficiency.Experimental results show the proposed algorithmcan perform better in a Grid environment.

General Terms
Algorithms, Performances, Experimentation

Keywords
Grid Computing, Job Scheduling, Load Balancing, Computational Grids, Resource Management.

1. INTRODUCTION
Grid computing is emerging as a new paradigm for solving complex scientific and engineering problems. Basically, it is a form of distributed computing that enables the users to share the widely distributed, heterogeneous resources connected through the network to carry out their complex computational tasks [4]. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1].There are many issues in using computational grid. How to appropriately and efficiently assign resources to tasks, generally called job scheduling, is one of the important issues. 'Scheduling' is "the processes of ordering tasks on compute resources and ordering communication between tasks" [2]. The main purpose of job scheduling is to shorten the job completion time and enhance the system throughput. A grid scheduling system should take the various characteristics of grid applications and resources into account. In a grid environment, the resource providersand tasks are all changing constantly, so the traditional scheduling algorithms, e.g. ''First Come, First Serve'' may not be suitable for a dynamic grid system. It is very important to assign appropriate resources to tasks. Through a good scheduling method, the system can perform better and applications can avoid unnecessary delays. Various algorithms [4,5,6] are proposed to schedule jobs in grid environments. Although many proposed scheduling algorithms proved that they are suitable for a dynamic environment, only little work has been done on the aspect of job scheduling considering the real time characteristics of grid resources.

Load balance is also an important issue in grid environment. The main purpose of load balancing is to balance the load of each resource in order to enhance the resource utilization and increase the system throughput. For a conventional distributed system, many load balancing algorithms [6–9] have been proposed. But they may not be suitable for grid environments due to the different characteristics in grids. Numbers of load balancing algorithmshave been proposed for grid environments. Some take the grid characteristics into account but do not follow changes in the system status. Others may set a fixed balance threshold for controlling the load situation of the whole grid system. Hence, they might not be suitable in a dynamic grid environment. Based on this opportunity for improvement, we propose a new framework and scheduling algorithm to balance the load of a grid system with an adaptive balance threshold while trying to minimize the makespan of job execution. We assign a job to a resource depending on the resource's characteristics while simultaneously considering the load of the cluster. Local and global updates allow the refreshment of the new status of resources in the grid system. A more appropriate scheduling is achieved via these updates.

This paper is organized as follows. Section 2 is an overview of related work about job scheduling in Grid environment. Our proposed grid framework and job scheduling algorithm are presented in Section 3. Section 4 contains description of Optimized Hierarchical Load Balancing Algorithm. Section 5 containsexperiment parameters, set-up, and results.Finally, Section 6 and 7givesConclusion and References.

2. RELATED WORK
In the literature, many scheduling algorithms have been proposed. Most of them can be applied to the grid environment with suitable modifications.
- First Come First Served scheduling algorithm (FCFS)

In this algorithm, jobs are executed according to the order of job arriving time. The next job will be executed in turn. The FCFS algorithm [4] may induce a ''convoy effect''. The convoy effect happens when there is a job with a large amount of workload in the job queue. When this occurs, all thejobsqueued behind it must wait a long time for the long job to finish.

- Round Robin scheduling algorithm (RR)

The RR algorithm [5] mainly focuses on the fairness problem. The RR algorithm defines a ring as its queue and also defines a fixed time quantum. Each job can be executed only within this quantum, and in turn. If the job cannot be completed in one quantum, it will return to the queue and wait for the next round. The major advantage of RR algorithm is that jobs are executed in turn and do not need to wait for the previous job completion. Therefore, it does not suffer from a starvation problem. However, if the job queue is fully loaded or workload is heavy, it will take a lot of time to complete all the jobs. Furthermore, a suitable time quantum is difficult to decide.

- Min–min and max–min algorithm

The Min–min scheduling algorithm [3] sets the jobs that can be completed earliest with the highest priority. Each job will always be assigned to the resource that can complete it earliest. Similar to Min–min algorithm, Max–min algorithm [3] sets the highest priority to the job with the maximum earliest completion time. The main idea of Max–min algorithm is to overlap long running tasks with short-running tasks.Max–min can be used in cases where there are many shorter tasks than there are longer tasks. For example, if there is only one long task, Min–min will first execute many short jobs concurrently, and then execute the long task. Max–min will execute short jobsconcurrently with the long job.

- Sufferage scheduling algorithm

The idea behind the sufferage scheduling algorithm is that better mapping can be generated by assigning a machine to a task that would ‘‘suffer’’ most in terms of expected completion time if that machine is not assigned to it. In this algorithm, each job is assigned according to its sufferage value. The sufferage value is defined as the difference between its second earliest completion time and its earliest completion time (two completion times with different resources). The sufferage algorithm will pick a job in an arbitrary order and assign it to the resource that gives the earliest completion time. If another job has the earliest completion time with same resource, the scheduler will compare their sufferage values and choose the larger one. However, this algorithm mayhave the starvation problem.

- Most Fit Task scheduling algorithm (MFTF)

The MFTF algorithm [7] mainly attempts to discover the fitness between tasks and resources for user. It assigns resources to tasks according to a fitness value, and the value is calculated as follows:

$$Fitness(i, j) = 10000/(1 + |W_i/S_j - E_i|) \quad (1)$$

where Wi is the workload of the i_{th} task, S_j is the CPU speed ofthe j_{th} node, and E_i is the expected time of the i_{th} task. W_i/S_j isthe expected execution time using this node.

$|W_i/S_j - E_i|$ is thedifference of the estimated execution time and the expected taskexecution time. E_i is determined by the user or estimated by themachine. How to set E_i is calculated by (2).

$$E_i = A + n \times S \quad (2)$$

where A is the average response time;n is a non-negative real number and S is the standard deviation oftask response time.When the estimated execution time is closer to E_i, it meansthat the node is more suitable for the task. However, the MFTF scheduling algorithm has some problems for estimating. It does not consider the resource utilization, and the estimated function is an ideal method. Therefore, incorrect scheduling may occur in the real environment.

- ACO algorithms in job scheduling

Ant Colony Optimization (ACO) [8] was used for solving the scheduling problem in grids in recent years. Xu et al.proposed a simple grid simulation architecture and modified the basic ant algorithm for job scheduling in grid. Thescheduling algorithm proposed in the paper needs some information such as the number of CPUs, Million Instructions Per Second (MIPS) of every CPU for job scheduling. A resource must submit the information mentioned above to the resource monitor.The pheromone for calculating resource suitability will be initialized as:

$$\tau j(0) = m * p + c/s \quad (3)$$

Where$s_j(0)$ means the initial pheromone of the path between the resource monitor and resource j, m is the number of CPUs, p is the MIPS of one CPU, c is the size of parameters, and s_j is the transfer time from resource j to the resource monitor.

Encourage-factor and punish-factor are added into the original ant algorithm. When resource j completes a job successfully, the pheromone will be updated a

$$\tau jnew = \rho.\tau jold + Ce * k \quad (4)$$

$\boldsymbol{\tau}_j^{new}$ is the pheromone after updating, $\boldsymbol{\tau}_j^{old}$ is the pheromone before updating, q is the permanence of pheromone, Ce isthe encourage factor, and K is the computing and transferring quality of the job.In contrast, when resource j does not complete the job, the pheromone will be updated by a punish-factor as:

$$\tau jnew = \rho.\tau jold + Cp * k \quad (5)$$

whereC_p is the punish-factor.

Encourage-factor and punish-factor are used to adjust the pheromone of every resource and select the better resource tosubmit jobs. However the load of the better resources will be more than others and it will decrease the performance of jobscheduling.

3. PROPOSED SYSTEM MODEL

3.1 SystemArchitecture

The System Architecture is composed of four main components:Portal, Information Server, Scheduler, and clusters withgrid resources, as shown in Fig.1.The Portal provides an interface for users to submit jobs. TheInformation Server discovers resource nodes registered with thesystem, and records the information of the resource such as CPUspeed, idle CPU percentage, memory utilization and average load ofeach cluster, etc. The job scheduler accepts the job from the portaland uses the OHLBA with theinformation from Information Serverto choose the appropriate cluster and compare its load with thesystem. Then, it selects the resource with the strongest computingpower in the cluster to execute the submitted job. After the job isfinished, the result and the new status of the resource will be sentback to the Information Server for another scheduling.

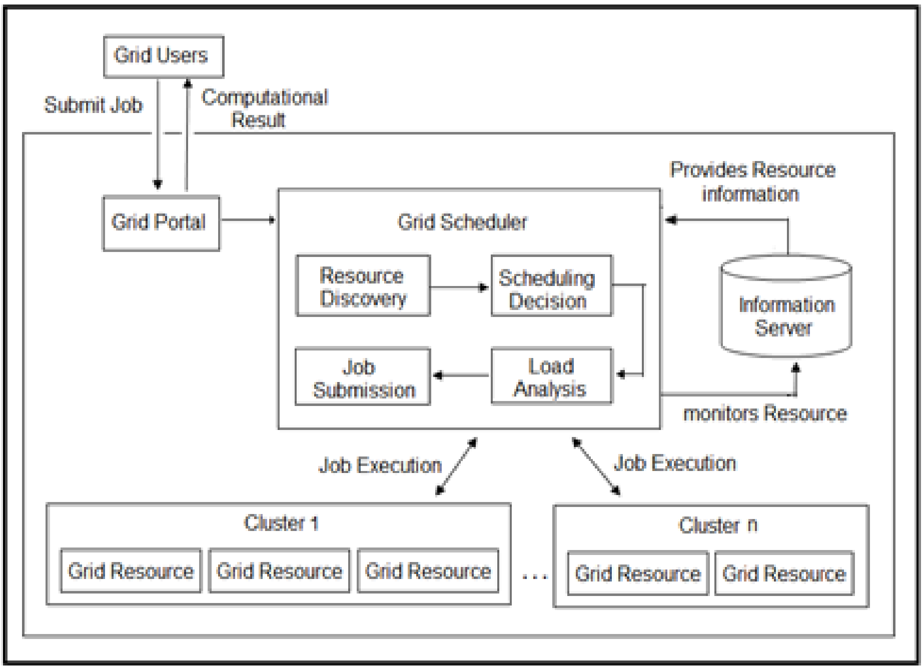

Fig 1: System Architecture

3.2 Proposed Job Scheduling Algorithm

Optimized Hierarchical Load Balancing Algorithm

Input: Given Job
Output: Computational result with less makespan.
Step 1: User submits jobs to portal.
Step 2: Scheduler Initialize all the parameters in table1.
Step 3: Scheduler obtains information such as ACP, ALC, AL,AD_{TH} from GIS.
Step 4: While job is not empty do
Select the job
Sort the ACL of clusters
if ACL <AD_{TH} then
cluster UNDERLOADED
Allocate job to resource
else
cluster OVERLOADED
Select the highest ACP_i for remaining clusters
Assign job to resource with highest ACP_i
Local Update
end if
repeat step 4
end While
Step 5: If all jobs are executed then
Global Update
else
Select a job
end if

3.3 Phases of Job Scheduling Algorithm

'Scheduling' is the process of ordering taskson compute resources and ordering communication between tasks known as the allocation of computation and communication over time.There are three main phases of Grid scheduling. Phase one is resource discovery, which provides a list of available resources. Phase two is resource allocation and load analysis, which involves the selection of feasible resources and the mappingof jobs to the resources. The third phase includes job execution.

3.3.1 Resource Discovery

Resource discovery is the initial phase of any scheduling algorithm. The resource discovery algorithm is shown in fig 2. This phase is carried out in two steps
Step 1: Resource Listing
The list of machines or resource to which the user has access to is listed from the resource pool which was created previously.
Step 2: Resource Filtering
Given a set of resources to which a user has access and the resources are filtered according to Average Computation Power of each clusters.

Algorithm for Resource Discovery Phase

Input: Client Job J_i , load, time limit, cost limit. Cluster of Resource R_i

1. Discover the available resources from inter domain and intra domain according to various load, capacity, PE and bandwidth
2. Find the Average Computational Power of each cluster and select the resource according to ACP value.

For every cluster C_i

- Calculate Average Computational Power (ACP) of cluster

$$ACP = \frac{\text{Total capacity} * \text{Available capacity}}{\text{no.of resources}} \quad (6)$$

- Sort the Cluster based on ACP value.
- Filter the resources from cluster according to the highest ACP and store it in list R_L
- Select the resources with highest ACP and add it in the list R_L

Fig 2: Resource Discovery Algorithm

3.3.2 Load Analysis

In this module the Average Load index of each resource and clusters are calculated. The average load of each resource is estimated by the weighted sum of squares method. To calculate the weighted sum of squares method, we use three parameters namely network utilization, memory utilization, and idle CPU percentage. The Load balancing algorithm is shown in fig 3.

Algorithm for Load Analysis

Begin
Get the resource R_L in the queue
For all (selected resources R_i in the cluster)

1. Find the average load of each resources (ALC) from $R_{L.}$

$$ALC = \frac{\sqrt{((\alpha*CPUk^2)+(\beta*MUk^2)+(\gamma*NUk^2))}}{total\ \ no.of\ resources} \quad (7)$$

Where,

CPU_k-- CPU Utilization
MU_k-- Memory Utilization
NU_k --- Network Utilization
α, β, γ -- Weighted values

2. Find the overall load of system(AL)

$$AL = \frac{ALC}{total\ \ no.of\ clusters} \quad (8)$$

End

Fig 3: Load Analysis Algorithm

3.3.3 Job Submission

Set an adaptive threshold value and check whether the Average computation power of cluster is less than threshold. If the value is less, then the cluster is underload. Scheduler submits the job to the cluster which is in underload. The job submission algorithm is shown in fig 4.

Algorithm for Job Submission

Begin

1. Assign Adaptive Threshold AD_{TH} as AL
   AD_{TH} =AL
2. Check whether the cluster is overloaded or underloaded
   if(ALC_i<AD_{TH}) {
   Cluster underloaded
   Assign the job to the cluster
   }
   else{
   Cluster overloaded
   Choose new cluster
3. Calculate the job completion time for the jobs submitted to the grid.

End

Fig 4: Job Submission Algorithm

4. OPTIMIZED HIERARCHICAL LOAD BALANCING ALGORITHM

Optimized Hierarchical Load Balancing Algorithm mainly focuses on the computational grid environment. When the scheduler receives a job submitted by a user, it will transfer a request to the Information Server in order to obtain the necessary information such as the idle CPU percentage of each resource, average load of each cluster and average load of the system. Then the scheduler chooses a cluster which has the fastest average computing power (ACP_i). The average computing power of the cluster is defined in equation(6). When a job is to be assigned to a cluster with the highest ACP_i, the load of the selected cluster will be checked first to see if it is already overloaded.After the scheduler selects the cluster which has the fastest ACP, it will compare the average load of the chosen cluster with the average load of the system. The average load of the cluster is defined by the average load of each resource in cluster i.To calculate the average load of each resource in OHLBA, we consider three load attributes, CPU utilization of the resource (CPU_k), the memory utilization of the resource (MU_k) and the utilization of network (NU_k). The average load of each cluster i (ALC_i) is defined in equation(7). Then calculate the average Load. The average load of the system (AL) is defined in equation(8). We set the average load of each cluster i, ALCi, to be less than the balance threshold of the system.

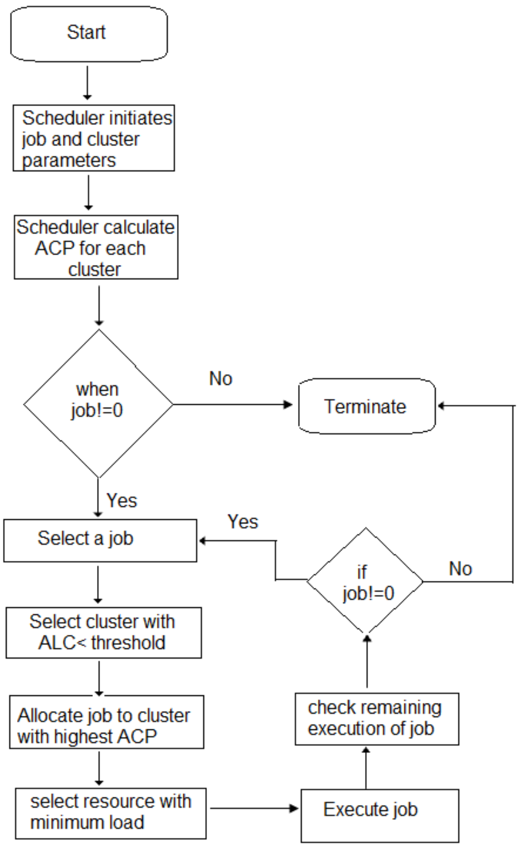

Fig. 5The process in OHLBA.

The process of OHLBA algorithm is shown in Fig 5. In our algorithm, when scheduler receives a job and obtains necessary information from the job, we will sort clusters by their average loads. If the average load of cluster (ALC) exceeds the balance threshold (AD_{TH}), it means that the cluster is overloaded. We sort the clusters which are underloaded and select the cluster with the highest ACP within those clusters. After selecting the suitable cluster, we select the resource with the best computing power in this cluster and assign the job. Local update and global update are also performed in OHLBA to ensure that we can get the latest status of resources.

5. SIMULATION AND RESULT ANALYSIS

The simulation was carried out on the excellent grid simulation toolkit GridSimToolKit 5.0 [10] 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.This section analyzes the performance of the scheduling strategy. The parameters used for simulation is shown in Table1.

Table1:Simulation parameters of our proposed algorithm

Parameter	Value
Number of tasks	2000
Size of task (MI)	300,000–500,000
Number of nodes of a cluster	10
Computing power of resource node (MIPS)	500–5000
Number of clusters	10
Size of memory (MB)	500–1000
Baud rate (bps)	500–1000
User submitted number of jobs	50

5.1 Results of different scheduling algorithms

The results of different Scheduling algorithms focuses on the makespan of jobs.Wecompare the result of our algorithm OHLBA with MFTF algorithm andACOalgorithm.

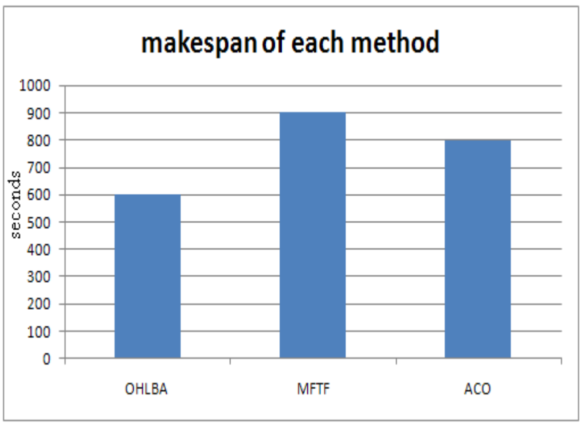

Fig 6.Makespan of each scheduling algorithms

According to Fig. 6, we can observe that OHLBA has betterperformance than other algorithms. We assign jobs to the resourcedepending on the status of the resource. A cluster with highestaverage computing power means that it is the suitable cluster forthe resource. The average computing power of each cluster will be calculated according to the newest status of resources which isupdated by local updateand global update. Therefore, OHLBA canselect the suitableresources for jobs and reduce the makespan.

6. CONCLUSION

Load balancing is one of the main issues in the grid environment. Recent researches have proved that loadbalancing on computational grids is best solved by heuristic approach. Hence, an Optimized Hierarchical Load Balancingalgorithm is developed to allocate tasks to proper resources. In order to verify the performance of proposedalgorithm, the simulation is performed. The results of the experiments are also presented and the strength of thealgorithm is investigated. The simulation result concludes that the proposed algorithm enhances performance interms of resource utilization.

7. REFERENCES

[1] Y. H. Lee, S. Leu and R. S. Chang, “Improving job scheduling algorithms in a grid environment”, Future generation computer systems, (2011)May.

[2] H. Shan, L. Oliker, W. Smith and R. Biswas, “Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration”, (2004).

[3] Ruay-Shiung Chang, Chih-Yuan Lin, Chun-Fu Lin, “An Adaptive Scoring Job Scheduling algorithm for grid computing”, Future Generation Computer Systems 207 (2012) 79–89.

[4] Ungurean, “Job Scheduling Algorithm based on Dynamic Management of Resources Provided by Grid Computing Systems”, Electronics and Electrical engineering, vol. 103, no. 7, (2010).

[5] Sheng-De Wang, I-Tar Hsu, Zheng-Yi Huang, “Dynamic scheduling methods forcomputational grid environments”, International Conference on Parallel andDistributed Systems 1 (2005) 22–28.

[6] P.K. Suri, Singh Manpreet, “An efficient decentralized load balancing algorithm for grid”, 2010 IEEE 2nd International Advance Computing Conference, IACC, 2010, pp. 10–13.

[7] S. Sharma, S. Singh and M. Sharma, “Performance Analysis of Load Balancing Algorithms”, WorldAcademy of Science, Engineering and Technology, vol. 38, (2008), pp. 269-272.

[8] R. S. Chang, C. F. lin and J. J. Chen, “Selecting the most fitting resource for task execution”, Future Generation Computer Systems, vol. 27, (2011), pp. 227-231.

[9] Ruay-Shiung Chang, Jih-Sheng Chang, Po-Sheng Lin, “An ant algorithm forbalanced job scheduling in grids”, Future Generation Computer Systems 25 (1)(2009) 20–27..

[10] R. Buyya and M. M. Murshed, “Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing”, Concurrency and Computation: Practice and Experience, 14:1175–1220, 2002.