AN ADAPTIVE ALGORITHM FOR TASK SCHEDULING FOR COMPUTATIONAL GRID

P.Keerthika[1], P.Suresh[2]

[1]Assistant Professor (Senior Grade), Department of CSE, Kongu Engineering College, Tamilnadu, India
[2]Assistant Professor (Senior Grade), Department of IT, Kongu Engineering College, Tamilnadu, India

Abstract

Grid Computing is a collection of computing and storage resources that are collected from multiple administrative domains. Grid resources can be applied to reach a common goal. Since computational grids enable the sharing and aggregation of a wide variety of geographically distributed computational resources, an effective task scheduling is vital for managing the tasks. Efficient scheduling algorithms are the need of the hour to achieve efficient utilization of the unused CPU cycles distributed geographically in various locations. The existing job scheduling algorithms in grid computing are mainly concentrated on the system's performance rather than the user satisfaction. This research work presents a new algorithm that mainly focuses on better meeting the deadlines of the statically available jobs as expected by the users. This algorithm also concentrates on the better utilization of the available heterogeneous resources.

***Keywords:** Task Scheduling, Computational Grid, Adaptive Scheduling and User Deadline.*

---***---

1. INTRODUCTION

Even with the emergence of many superfast computers and the high speed networks, the utilization of the geographically distributed resources has gained huge importance. This recognition is mainly because of the low cost services and the best outcome offered by them. There are many computing fields that offer high utilization of the widely available underutilized resources such as grid computing, distributed computing, parallel computing, etc. Grid computing has gained more popularity because of it loosely coupled nature when compared to distributed computing and parallel computing that mainly deals with tightly coupled systems. The basic idea of grid Computing is to utilize the ideal CPU cycles and storage of millions of computer systems across a worldwide network function as a flexible, pervasive, and inexpensive accessible pool that could be harnessed by anyone who needs it, similar to the way power companies and their users share the electrical grid [1].

When considering the scheduling of the resources many factors such as CPU utilization rate, throughput, turnaround time, waiting time, response time should be focused for all the processors when assigned with the jobs. Thus the jobs are assigned to the resources considering the system's performance. Thus the scheduling plays an important role in achieving the best utilization of resources and the better completion of the submitted jobs. Many scheduling algorithms have been designed for this purpose. Even then scheduling is a main focus. There are many algorithms that are mainly system centric i.e. consider the effective utilization of resources such as MCT, MET, OLB, Min-min, Max-min, First Come First Serve (FCFS) Algorithm, Shortest Job Fastest Resource (SJFR), Longest Job Fastest Resource (LJFR), etc... But these traditional algorithms mainly focus on the system performance of the user expected time for each job. In this paper we have proposed a new idea that considers the time expected to complete the job by the user and schedules the job by concentrating on both the system performance and the user satisfaction.

Then, after scheduling the jobs through our algorithm, we can guarantee that each job will be assigned to their most suitable resources, and most of the jobs are completed within their respective requisition times, thus the satisfaction of the users is demonstrated. Then in the algorithm of this paper, we take into consideration as how to satisfy the demands of the users as much as possible without decreasing system performance much are just the main focus. And we compare the efficiency of our algorithm with the conventional algorithms.

2. RELATED WORKS

Grid scheduling is the process of scheduling jobs over widely distributed grid resources. This is achieved by using a grid scheduler. A grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and usually owns the resource [2]. There are three generalized stages in the scheduling procedure of the Grid computing. They are resource discovering and filtering, resource allocation and scheduling according to certain strategies and job submission and job execution management over multiple administrative domains. As this paper mainly concentrates on the job scheduling we need to concentrate on the second stage. Scheduling algorithms can be divided into two major modes one is static scheduling and the other one is dynamic scheduling, with each having their own advantages and disadvantages.

In the case of dynamic mode, we don’t do the advanced estimate of jobs before assigning the jobs to the resource. Then we monitor the resources available from time to time and reschedule and reassign the jobs to the best resource available to improve the system’s performance. We also have to make some necessary adjustments for reassigning the jobs to the best resource such as rescheduling and transferring jobs to most suitable resources. In this paper, we only take static mode into consideration.

In static mode, a prior estimate of jobs with the available resources are done and assigned to the suitable resource based on some strategy. After the submission of the job to a resource no change will happen to this allocation dynamically. One of the major advantage of this mode is this is easy and simple to carry out.

First Come First Serve (FCFS) algorithm schedules the job to the available resources linearly without considering any parameters of the job and resource. Shortest Job Fastest Resource (SJFR) scheduling algorithm mainly aims to reduce the makespan of all the submitted jobs. The makespan is the total time taken by all the statically available jobs to get executed on their scheduled resources. The shortest job is first scheduled and allocates the jobs to the fastest resource available [2].

Longest Job Fastest Resource (LJFR) is a scheduling algorithm, which also tries to reduce the makespan of the scheduled jobs. The longest job is scheduled to the fastest resource. Since the longest job is submitted to the fastest resource the execution time of the longest job is drastically reduced when compared to its execution time on any other resource in the grid. As far as execution time is considered it gives the best results [3].

Opportunistic Load Balancing (OLB) assigns each task, in arbitrary order, to the next machine that is expected to be available, regardless of the task's expected execution time on that machine. The intuition behind OLB is to keep all machines as busy as possible. The advantage of OLB is its simplicity. The major drawback of it is, it does not consider the characteristics of job and resource which results in very poor makespan [4].

In contrast to OLB, Minimum Execution Time (MET) assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability. The motivation behind MET is to give each task to its best machine. It causes severe load imbalance across machines [4].

Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task. This causes some tasks to be assigned to machines that do not have the minimum execution time for them. The intuition behind MCT is to combine the benefits of OLB and MET and hence to improve the makespan. But all the above mentioned algorithms deals with only one job at each mapping time and hence it is not more suitable for the heterogeneous environment [4].

The Min-min scheduling algorithm deals with the set of all unscheduled independent tasks. Then the ETC (Expected Time to Compute) matrix will be computed. ETC matrix contains the estimated completion time for all the jobs in every available resources. Then for each job the minimum completion time will be selected from the ETC matrix. Then the overall minimum value will be chosen from the minimum completion time of all unmapped jobs and is assigned with its best resource. Then the ETC matrix is recalculated for the remaining unmapped jobs and the above process is repeated until all the jobs are assigned with the resources. Min-min maps the tasks in the order that changes the machine availability status by the least amount that any assignment could. When compared to the above mentioned algorithms Min-min has the best makespan. Here, mapping the task with the shorter execution time to its best machine allows the tasks to be completed very soon [4].

The Max-min scheduling algorithm is very similar to Min-min. The Max-min scheduling algorithm also deals with the set of all unscheduled independent tasks. Then the ETC (Expected Time to Compute) matrix will be computed for all the unmapped jobs in every available resource. Then for each job the minimum completion time will be selected from the ETC matrix. Then the overall maximum value will be chosen from the minimum completion time of all unmapped jobs and is assigned with its best resource. Then the ETC matrix is recalculated for the remaining unmapped jobs and the above process is repeated until all the jobs are assigned with the resources. Here, mapping the task with the longer execution time to its best machine first allows this task to be executed concurrently with the remaining tasks. And hence makespan is improved [5].

As mentioned above, Min-min algorithm is more advantageous in grid scheduling when compared with other algorithms. Even though it concentrates on the system performance the user satisfaction is not taken into consideration. So we need to modify this aspect and provide an algorithm that takes both the user satisfaction and system performance into account. Here we are going to discuss a new scheduling strategy that guarantees user’s deadline to be satisfied through our new scheduling strategy without sacrificing the system’s performance [6].

3. PROPOSED SCHEDULING MODEL

Grid scheduling is the process of scheduling application tasks over grid resources. There are two main concepts in this scheduling process namely system’s performance and user satisfaction. Essentially, all the conventional algorithms mainly concentrate on the system’s performance. As these algorithms focus on the proper utilization of the computational power of the grid resources, these are referred to us as the system-centric algorithms.

The application-centric algorithms chiefly focus on the contentment of the demands i.e. deadlines of the independent tasks provided by the corresponding users. Here the satisfaction of the users is mainly concentrated rather than focusing on the performance characteristics of the grid resources.

This proposed algorithm mainly deals with the statically available jobs and hence it is of static scheduling mode. In particular this algorithm deals with a list of jobs at a time and has two phases in scheduling such as task prioritizing and resource selection. In the task prioritizing phase sets the priority of each task with the user deadline as the parameter and generates a scheduling list by sorting the tasks according to their rank values. The resource selection phase selects tasks in the order of their priorities and maps each selected task on its optimal resource. So our algorithm falls into list scheduling algorithms. This list scheduling is further classified into batch mode, dependency mode and dependency-batch mode.

Notation	Definition
CTi,j	Completion time of the job or task Ji in the resource rj
RTj	Ready time of the resource rj
ETi,j	Execution time of the job or task Ji in the resource rj
DCTi,j	Difference in time between the deadline given by the user and the calculated completion time for the job in available resources
MinDCTi,j	The minimum value from the difference values DCTi,j for the given job
UTi	User requisition time or the deadline given by the user for the jobs in U

Batch mode scheduling algorithms are initially designed for scheduling parallel independent tasks, such as bag of tasks and parameter tasks, on a pool of resources. Since the number of resources is much less than the number of tasks, the tasks need to be scheduled on the resources in a certain order. A batch mode algorithm intends to provide a strategy to order and map these parallel tasks on the resources, in order to complete the execution of these parallel tasks at earliest time [7]. Even though batch mode scheduling algorithms aim at the scheduling problem of independent tasks; they can also be applied to optimize the execution time of the submitted jobs which consists of a lot of independent parallel tasks with a limited number of resources.

Dependency mode scheduling algorithms are derived from the algorithms for scheduling a task graph with interdependent tasks on distributed computing environments [8]. It intends to provide a strategy to order and map the submitted tasks on heterogeneous resources based on analyzing the dependencies of the entire task graph, in order to complete these interdependent tasks at earliest time. Unlike batch mode algorithms, it ranks the priorities of all tasks in the submitted jobs at one time.

Dependency-batch mode scheduling algorithms combine dependency mode and batch mode. It first assigns the rank to the jobs like that of the Batch mode scheduling algorithm and then it adapts dependency mode scheduling algorithm to schedule the independent tasks within the submitted jobs. As this algorithm deals with the independent tasks, the first step involved in it is the assignment of rank to the tasks on considering the deadline of the user as the parameter. The next step involved in it is the resource selection and hence our algorithm falls into the batch mode scheduling algorithm.

Let us consider the mathematical representations to denote the relationships between the resources and jobs. And also to introduce the parameters the parameters involved in our algorithm such as execution time, completion time, ready time, etc. that have been used in our algorithm. The resource set is represented as P= { r1, r2, r3,………., rm}. As the grid environment deals with the heterogeneously distributed grid resources the number of resources available may be huge. As we consider the static environment both the jobs submitted and the resource available are taken as fixed and they do not change over time.

The jobs submitted can be enclosed within the job set which is represented as U= {J1, J2, J3,…………,, J4}. The jobs submitted are considered as the independent tasks that can be executed in parallel with other available tasks. Also the jobs are considered as static i.e. they number of tasks submitted are fixed and they do not change with time. The users submitting their jobs for execution are represented as C= {C1, C2, C3, ……………., Cp}. The users submit the jobs with the requisition time i.e. within which the job needs to be completed which can also be called the demanded deadline of the user for the submitted jobs.

4. PROPOSED ALGORITHM

ETC matrix is constructed for all the jobs with every available resource. Secondly, the job with the minimum deadline is considered. The deadline of the selected job given by the user is compared with different ETC values. The job is allocated to the resource that has the minimum difference value. Then, the job is removed from the job set. Next, the waiting time of the resource is changed and the ETC matrix is recalculated for the remaining unmapped jobs. Above steps are repeated until all the jobs are scheduled.

4.1 Calculation of ETC Matrix

ETC matrix is basically the execution time matrix. A prior estimation of the execution time of the submitted jobs in every available resource is computed by considering the characteristics of the tasks and resources. In the proposed system, 512 tasks and 16 resources are considered. Based on the characteristics, the tasks can be classified into High Task and Low Task. Similarly the resources can be classified into High Machine and Low Machine

By considering the various characteristics of the job and resource into account, ETC matrix can be designed and it can be classified into Consistent, Inconsistent and Partially-consistent. An ETC matrix is said to be consistent, if a resource Ri executes a task Ti faster than the resource Rk and Ri executes all other jobs faster than Rk. For a matrix to be inconsistent, a resource Ri executes some jobs faster than Rj and some jobs are slower than Rj. Partially-consistent ETC matrices are also called as semi-consistent matrices. A semi consistent matrix is a sub matrix of inconsistent matrix with predefined size.

The proposed algorithm is mainly based on user satisfaction and system performance. It takes user's deadlines into account and makes the job to be executed within the expected deadline by assigning it to the suitable resource. It also concentrates on the system performance by reducing the idle time of the resources and assigning the tasks equally among the available resources. It considers the ETC matrix and concentrates on the completion time and hence the system's performance is also the major consideration in addition to user's satisfaction. The proposed scheduling process is performed as two major steps. In the first step, we concentrate on the user satisfaction and in the second step we consider system performance. Firstly the ETC matrix is constructed for the available resources with every available resource. Secondly we consider the job with the minimum deadline i.e. the job that needs to be completed quickly. Then the deadline of the selected job given by the user is compared with that of different ETC values. Then allocate the job to the resource that has the minimum difference value. Then remove the job from the job set. Then the waiting time of the resource is changed and the ETC matrix is recalculated for the remaining unmapped jobs. Then continue the above steps until all the jobs are scheduled. Thus both the user satisfaction and system performance can be taken into consideration effectively with this algorithm.

In this algorithm, ETC matrix is constructed. Completion matrix is calculated by adding the ETC values with the ready time of the resources. With every job submitted the deadline is acquired as input within which the user expects the job to be completed. Priority is assigned to the jobs based on the deadline provided by the user. Initially, the job with the highest priority is chosen for scheduling.

The difference matrix is computed for the job that has been chosen. The minimum value is found from the difference matrix calculated. The job is allocated to the corresponding resource. Next step is the computation of the ready time and the execution time of that resource. The completion matrix is re-computed by adding the ready time of the resource. The steps from prioritizing the jobs are repeated until all the submitted jobs are scheduled to their most suitable resources.

5. RESULTS AND DISCUSSION

Adaptive job scheduling algorithm is compared with Min-min algorithm and application demand aware scheduling algorithm. The parameters considered for the comparison are

- Makespan- the total time taken to complete all the submitted jobs
- Hit - a task is said to be a hit if it is completed within the user requisition time.
- Miss - a task is said to be a miss if it is not completed within the user deadline.

Adaptive job scheduling algorithm has a better makespan when compared with the application demand aware scheduling algorithm. Also, the hit is high when compared with the Min-min algorithm. Even though the application demand aware scheduling algorithm has a high hit when compared to the Min-min algorithm, the makespan is high for the application demand aware scheduling algorithm. The adaptive job scheduling algorithm has an improved hit when compared to Min-min and better makespan when compared to application demand aware scheduling algorithm.

The various characteristics of ETC matrices (i.e. Consistent, Inconsistent, Partial), diverse tasks (such as high and low tasks) and variety of machines (such as high and low machine) are considered for the construction of twelve different ETC matrices. The application demand aware algorithm and the adaptive job scheduling algorithm are compared in terms of makespan, hit and miss.

The Application demand aware algorithm and the proposed algorithm is compared based on makespan and the values are given in table 1. The performance analysis is given in figure 1.

Table 1. Makespan Values

Tasks and Resources	**Application Demand Aware**	**Adaptive Job Scheduling**
High-High Partial (p-hh)	5774893	4504474
High-Low Partial (p-hl)	1470189	546691
Low-High Partial (p-lh)	106230	60084
Low-Low Partial (p-ll)	15524	13854
High-High Inconsistent (i-hh)	6804441	5403760
High-Low Inconsistent (i-hl)	1061204	855583
Low-High Inconsistent (i-lh)	145245	122041
Low-Low Inconsistent (i-ll)	10591	5429
High-High Consistent (c-hh)	9618108	7308179
High-Low Consistent (c-hl)	735913	469402
Low-High Consistent (c-lh)	84511	58645
Low-Low Consistent (c-ll)	11583	4947

The comparison based on number of hit is given in table 2 and figure 2. The performance of the proposed algorithm is better when compared to the application demand aware algorithm.

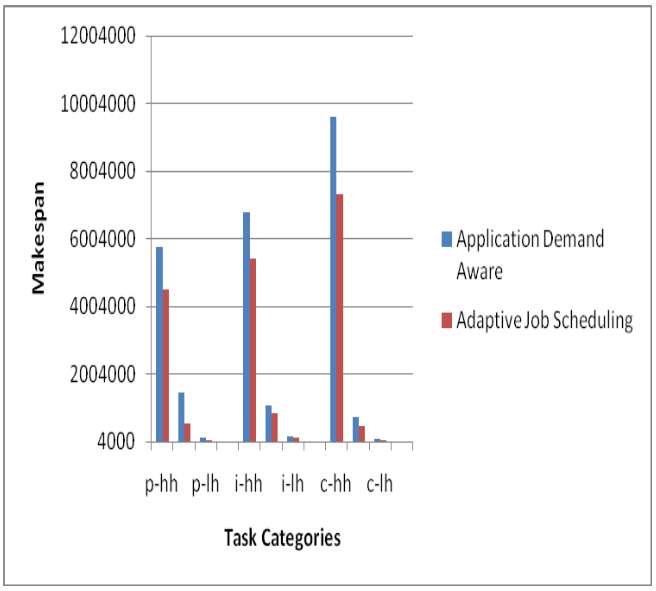

Fig.1 Comparison based on Makespan

Table 2. Hit Count Values

Tasks and Resources	**Application Demand Aware**	**Adaptive Job Scheduling**
High-High Partial	316	490
High-Low Partial	145	226
Low-High Partial	174	251
Low-Low Partial	208	330
High-High Inconsistent	230	481
High-Low Inconsistent	120	220
Low-High Inconsistent	160	242
Low-Low Inconsistent	214	311
High-High Consistent	360	480
High-Low Consistent	116	217
Low-High Consistent	176	209
Low-Low Consistent	211	350

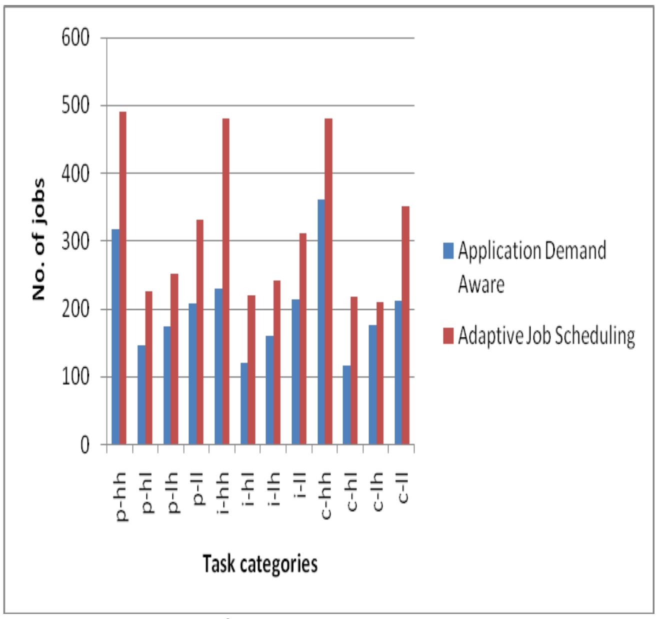

Fig.2 Performance based on Hit Count

6. CONCLUSION

Even though many scheduling algorithms have emerged to effectively utilize the globally available grid resources, they mainly concentrate on resource performance. They failed to contribute to the user's satisfaction. So, we focused to overcome the existing drawback with the help of our newly proposed algorithm. For each job, the user gives user requisition time and with the help of this we make sure that most jobs are completed within the deadline requested by the user. The user satisfaction is the major consideration of our idea, still the system performance is also preserved to a greater extent. This algorithm is more beneficial with respect to the user satisfaction when compared with Min-min algorithm and has good system performance when compared with the application demand aware scheduling algorithm. In the proposed system, only the independent tasks are considered. So this work can be extended to suite for dependent tasks. The next task is to study the function and feasibility of dynamic scheduling, to make some possible improvements on the current scheduling algorithm, hoping to make it more flexible and efficient in actual application.

REFERENCES

[1]. Aysan Rasooli, Mohammad Mirza-Aghatabar and Siavash Khorsandi (2008) 'Introduction of Novel Rule Based Algorithms for Scheduling in Grid Computing Systems', Second Asia International Conference on Modelling & Simulation, pp.138-143.

[2]. G.Sumathi and N.P. Gopalan, "Priority based scheduling for heterogeneous grid environments",1-4244-0411-8/06/$20.00 C 2006 IEEE.

[3]. Zhan Gao, Siwei Luo and Ding Ding, "A Scheduling Mechanism Considering Simultaneous Running of Grid Tasks and Local Tasks in the Computational Grid", 2007 International Conference on Multimedia and Ubiquitous Engineering(MUE'07) 0-7695-2777-9/07 $20.00 © 2007.

[4]. Tracy D. Braun, Howard Jay Siegel,2 and Noah Beck,” A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07) 0-7695-2777-9/07 $20.00 © 2007.
[5]. Y. Zhu, "A Survey on Grid Scheduling Systems", Department of Computer Science, Hong Kong University of science and Technology, 2003.
[6]. Sumathi. G., Gopalan. N.P, "Grid Scheduling Algorithms for Heterogeneous Environment", Proceedings of the IEEE Int. Conf. on Signal & Image Technology and Internet Based Systems (SITIS 2005), Cameroon, West Africa.
[7]. Hao Y, Liu G. and Wenc N, An enhanced load balancing mechanism based on deadline control on GridSim, Future Generation Computer Systems, Vol.28, 2012, pp. 657-665.
[8]. Homer Wu, Chong-Yen Lee , Wuu-Yee Chen, Tsang-Yean Lee, "A Job Schedule Model Based on Grid Environment ", Proceedings of the First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'7695-2823-6/07 $20.00 © 2007