AN ENHANCED ADAPTIVE SCORING JOB SCHEDULING ALGORITHM WITH REPLICATION STRATEGY IN GRID ENVIRONMENT

S. K. Aparnaa[1], K. Kousalya[2]

[1]*Student, Kongu Engineering College, Perundurai, Erode-638052*
[2]*Faculty, Kongu Engineering College, Perundurai, Erode-638052*

Abstract

Grid computing is a form of distributed computing that involves coordinating and sharing data storage and network resource. The goal of grid job scheduling is to achieve high system throughput and match the job to the appropriate available computing resource. The complexity of scheduling problem increases with heterogeneous nature of grid and is highly difficult to schedule effectively. Existing algorithm does not adapt to the dynamic grid environment. In order to utilize the power of grid completely and to assign job to the resource dynamically an efficient algorithm called Adaptive Scoring Job Scheduling (ASJS) was introduced. However the bandwidth and storage capacity occupied by data intensive and computational intensive job is high and each time the user have to specify whether the job is computational intensive or data intensive. . Due to this problem the jobs are not completed in time. To provide a solution to that problem Enhanced Adaptive Scoring Job scheduling algorithm is introduced. The jobs are identified whether it is data intensive or computational intensive and based on that the jobs are scheduled. The jobs are allocated by computing Cluster Score (CS). The jobs that are submitted by the user is divided into sub tasks and replicated. By using this strategy the job occupies lower storage capacity and bandwidth. Due to the dynamic nature of grid environment, each time the status of the resources changes and each time the Cluster Score (CS) is computed and the jobs are replicated and allocated to the most appropriate resources.

Keywords: *Grid Computing, Resources, Scheduling. Replication*

----------***----------

1. INTRODUCTION

Grid can be classified as computational grid and data grid. The computational grid facilitates efficient computation power and CPU available. The data grid facilitates efficient storage and distributions of data. A computational grid is a collection of heterogeneous computing nodes for computation intensive jobs. A data grid connects geographically distributed computer and storage resources, enabling users to share data and other resources.

Grid computing aims at aggregating resources such as Central Processing Unit (CPU) speed, load and storage space to solve a single task. Grid computing combines the power of both parallel computing and distributed computing. Distributed computing supports resource sharing and parallel computing supports computing power.

Scheduling is defined as the process of allocating jobs by selecting best resource from collection of resources. The main purpose of scheduling is to balance the entire system and complete the execution of jobs as soon as possible. In grid, many users may face hundreds of computers to utilize and it is impossible for anyone to assign jobs manually in grids.

A good job scheduling algorithm should adjust according to the changing the status of the entire environment and types of jobs. Some characteristics that are intrinsic to grids should be considered during scheduling, such as resources, dynamic nature of machines, network load, bandwidth latency and topology.

Data replication is an important optimization step to manage large data by replicating data in various sites. The major challenge is a decision problem i.e. how many replicas should be created and where replicas should be stored. Hence new methods are needed to create replicas that increase availability without using unnecessary storage and bandwidth.

2. RELATED WORK

Paranhos et al (2003) [2] presented algorithm to increase the performance of the system and to schedule Bag-of-Tasks (BOT) application. Workqueue with Replication (WQR) algorithm is used to solve the problem. The tasks are replicated the replica which completes the job first is considered as a valid execution of task and other replicas are cancelled. Although the performance of the system is increased it wastes a lot of computing power.

Sheng et al (2005) [9] proposed an adaptive and dynamic scheduling method called the Most Fit Task First Scheduling (MFTF) for a class of computational grids, which are characterized by heterogeneous computing nodes and dynamic task arrivals. This method calculates the expected task execution time in order to know which task is suitable for the node. The most suitable resource to a task is assigned based on the fitness value. The larger the fitness value will indicate the greater suitability between the task and node. Therefore, the load of the resources with greater speed becomes heavy and more number of tasks will be queued.

Yan et al (2005) [3] proposed an algorithm called improved ant algorithm for job scheduling in grid computing. This algorithm is based on an adaptive scheduling heuristics and load balancing component to improve the job finishing rate and load balancing rate. The load balancing rate which is related to job finishing rate is introduced to change the pheromone. This will make the job finishing rate for different resource being similar and the load balancing will be improved.

Figueria and Tan Trieu (2008) dealt with storage capacity planning for data grids. Due to massive size of data the task of managing and distributing data quickly is a problem and hence the planning of storage capacity in data grids are carried out. The storage capacity for each node is minimum and this is assigned based on the network topology and workload characteristics. The data replication strategies are used to reduce the bandwidth consumption. The dynamic replication strategy replicates file based on the notion that recently accessed files are more likely to be accessed again by the nearby nodes.

Chang et al (2009) [5] dealt with an algorithm called Balanced Ant Colony Optimization(BACO) algorithm which is used to balance the load of the entire system and to minimize the makespan time for the given set of jobs. The pheromone indicator for each resource and each job is calculated based the estimated transmission time and execution time. The jobs are assigned to each resource based on pheromone indicator and executes the job. This algorithm does not consider the real status of the resource it assigns resource not only to good performance but also to bad ones.

Abdi et al (2010) [1] dealt with an algorithm called Hierarchical Replication Strategy (HRS) to improve data access and efficiencies. The bandwidth between the regions are considered as the main factor replica selection and deletion. The Replica Manager controls the data which is transferred in each region. This algorithm reduces the execution time of the job by reducing the job access time.

Syed et al (2012) [6] proposed an algorithm called Dynamic Multilevel Hybrid Scheduling Algorithm(MHS) using Median and Dynamic Multilevel Hybrid Scheduling Algorithm using Square Root(MHR) to provide solution to fixed quantum problem. The main aim of this algorithm is to execute the jobs with minimum turnaround time and with minimum response time. The following are the features of this algorithm. First, this algorithm favours the shortest job for execution. Second, execute the job on the basis of a dynamic time quantum, to fairly distribute processor time among grid jobs. And third feature is that they always execute the longest job, thus avoiding starvation.

Wei et al (2012) [9] dealt with an algorithm to improve the grid task scheduling called as an improved ant algorithm. This algorithm focuses on task scheduling in unsuccessful situations, improve robustness and successful probability of task allocation to the resource and shorten the completion time of the task. The task finds the resource by using resource selection rule and if the task find the resource but fails to complete then using redistribution rule the tasks are again redistributed.

Chang et al (2012) [4] dealt with an algorithm called Adaptive Scoring Job Scheduling (ASJS) algorithm to shorten the completion time and enhance the system throughput. Both the data intensive jobs and computation intensive jobs are assigned to the cluster. The jobs are scheduled based the selection of the most appropriate resource. The jobs are allocated to the cluster which has the highest Cluster Score (CS) and the highest the cluster which has the highest computation power. Each time the status of the resource changes the global and local update rules are performed. Based on the global and local update the jobs are scheduled.

3. PROPOSED WORK

The proposed algorithm computes the Cluster Score (CS), the jobs that are submitted by the user is divided into sub tasks and replicated. The replicated tasks are allocated to the cluster which has the highest score.

3.1 Proposed Scheduling Architecture

In the proposed system, the user submits the job to the grid scheduler and based on the Cluster Score(CS) which is computed jobs are allocated to the cluster of resource. The grid scheduler identifies whether it is a data intensive job or computational intensive job by using job information specified by the user. The computational intensive job needs more computing power, the CPU_Available value will be large compared to other jobs and data intensive job needs more transmission power. These jobs need more bandwidth to transmit the job. Based on these conditions the jobs are identified and they are allocated.

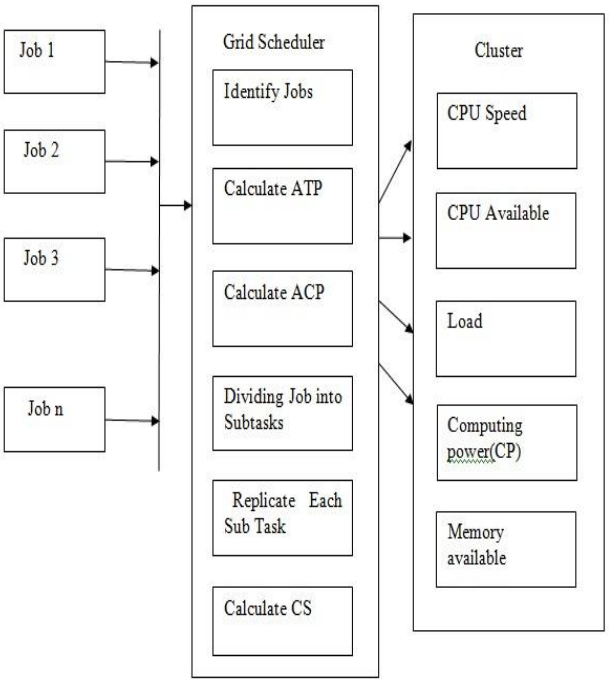

Fig 1: Overall architecture diagram

The proposed architecture which is illustrated in Fig 1 shows how jobs are allocated to the cluster of resources. Job 1, Job 2…Job n are the 'n' number of jobs which can be a computational intensive or data intensive jobs. The grid scheduler allocates the job to the cluster of resource by computing ATP,ACP. Suppose if the storage capacity and bandwidth need for the job is high the job is divided in sub task.Each subtask is replicated The cluster of resources include CPU Speed, Load, CPU Available, Computation power (CP) and Memory Available.

3.2 Enhanced Adaptive Scoring Job Scheduling Algorithm with Replication Strategy

In Enhanced Adaptive Scoring Job Scheduling algorithm along with ATP (Average Transmission Power) and ACP (Average Computation Power) [4] of each resource is calculated and the total size of the jobs are divided and the sub tasks are replicated. The replicated jobs are then assigned to the cluster which has the highest cluster score. The following algorithm gives a detailed description about how jobs are scheduled.

Step 1.The Average Transmission Power (ATP) which is calculated using the formula

$$\mathbf{ATP} = \sum_{j=1}^{m} Bandwidth\_available / m - 1$$

Bandwidth_available - bandwidth which is available between each cluster
m - number of clusters

Step 2 The Average Computation Power (ACP) which is calculated using the formula

$$\mathbf{ACP} = \sum_{k=1}^{n} CPU\_Speed * (1 - load) / n$$

CPU_Speed - speed of the resource in a cluster
load - current load of the cluster
n - number of resources in the cluster

Step 3 Let Sj be the total size of the job which is divided into subtasks

$$\mathbf{Sj = \{S_1, S_2 \ldots\ldots\ldots S_n\}}$$

S-Subtasks
n-number of subtasks

Step 4 For each sub task Sj the replica of the tasks are generated

$$\mathbf{Rj = \{RF_1, RF_2 \ldots\ldots\ldots RF_n\}}$$

RF-Replicated file
n-number of replicated file

Step 5 The Cluster Score (CS) is computed by using the formula

$$\mathbf{CS = \alpha.ATP + \beta.ACP}$$

α-coefficient value of ATP
β-coefficient value of ACP
The coefficient values should always be equal to1 ie, α+β=1

Step 6 The replicated tasks are allocated to the Cluster which has the highest Cluster Score (CS)

Step 7: The replica which finishes the job first is considered and other replicas are terminated.

The CP value is also calculated by taking the CPU Available value and dividing the value by 10 and the normalized value is divided with other CPU Available value to get CP value.

For example, 2 jobs Job1 and Job2 .Let Job1 be a data intensive job whose size is 200 MB and Job 2 be a computational intensive job. The Table 1 shows the status of the resource. The bandwidth between the Cluster A and Cluster B is 8.57 and bandwidth between Cluster B and Cluster C is 9.43 and bandwidth between Cluster A and

Cluster C is 10.57. Assume value of α=0.5, β=0.5 such that the sum of their values is equal to 1. Based on the job information which is specified by the user the values of α, β In the Table 1 the CPU Available which is calculated by using the formula
In the Table 1 the CPU Available which is calculated by using the formula

CPU_Available=CPU_Speed*(1-load)

Similarly the CP value is also calculated by taking the highest CPU Available value. The normalization range is set from 1 to 10.

Each time the status of the resource changes and based on the CPU Available and load information of the cluster the CP is calculated. The Job1 whose size is 200 MB is divided into subtask. Each subtask is replicated as RF_1,RF_2….RFn are allocate to the cluster. The replica which finishes the job first is considered and other replicas are terminated.

Based on the status of the resource given in Table 1 the formulas are applied to obtain the computation result for ATP, ACP,CS which is given in Table 2

The initial status of the resources which are considered here is R1,R2,R3,R4,R5,R6 and there are three clusters Cluster A, Cluster B and Cluster C. There are two resources that are grouped under each cluster. The resources that are considered are CPU Speed, Load, CPU Available and Memory Available.

The resources that are grouped under each cluster need not have to be uniform. Each cluster can contain multiple number of resources. Based on which the jobs are scheduled.

The replicas are allocated to the cluster and the replica which finishes the job first is considered and other replicas are terminated.

Table 1 Initial status of the resource

	Cluster A		**Cluster B**		**Cluster C**	
	R1	**R2**	**R3**	**R4**	**R5**	**R6**
CPU Speed (MHz)	3200	3000	3100	3100	3400	2800
Load(%)	30	25	25	20	25	20
CPU Avail (MHz)	2240	2250	2325	2480	2550	2240
CP	8.78	8.82	9.12	9.73	10	8.78

Based on the initial status of the resource the ATP, ACP values are calculated.

Table 2 Result of Computation after assigning Job 1

	Cluster A	**Cluster B**	**Cluster C**
ATP	9	9.57	10
ACP	8	9.43	9.37
CS	8.50	9.64	9.45

The Cluster B has the highest Cluster Score (CS) but the size of the job is 200 MB and the memory available in Cluster B is 20 MB hence the jobs are divided into tasks S1,S2…Sn and each subtask is replicated RF1,RF2..RFn. The replicas are allocated to the cluster. The replica which finishes the job first is considered and other replicas are terminated. Similarly for n number of jobs the CS is computed and allocated. The make span of n number of jobs are computed.

3.3 Algorithm Implementation

The following are the modules that are used for the implementation of the algorithm

```
Begin
Get the resources CPU_Speed, load,
CPU_Available,Memory_Available
Filter Job based on the job data given by the user
Get the number of clusters C1,C2… .Cn
For all cluster of resources
        Calculate ATP and ACP
Compute the Cluster Score (CS) for each cluster C1,C2..Cn
If (Size of the job > Memory_available)
{
  Divide the jobs into sub tasks S1,S2…….Sn
Each subtask Sj is replicated as Rj={RF1,RF2… .RFn}
Allocate replica Rj to the cluster
The replica which finishes the job first is considered
Other replicas are terminated
}
else
Select the Cluster based on Cluster Score(CS)
Check the CP of the Selected Cluster
Allocate job to the cluster which has the highest CP
After Assigning job status of the resource changes
Compute ATP, ACP, ASC, ALC and CS
Choose new cluster
Calculate the makespan time of the jobs submitted to grid
End
```

Fig.2 Algorithm Implementation

4. SIMULATION ENVIRONMENT

The simulation tool which is used for the proposed system is GridSim. The GridSim simulator provides a platform and enable the users to model and simulate the characteristic of grid resource. It supports job scheduling and distributes diverse set of resources for job scheduling. GridSim is a Java-based toolkit.

5. RESULTS AND ANALYSIS

The simulation parameters that are used for our proposed algorithm are

Parameter Value	
Number of Jobs	1000
Number of Resources	250
Computing Power	1500-5000
Number of Clusters	9
Size of memory(MB)	500
Jobs Submitted	100-200

Based on the above simulation parameter the ATP, ACP and CS values are calculated. The jobs are replicated and allocated to the cluster. The replica which finishes the job first is considered and other replicas are terminated.

Table 3: Makespan of jobs

Number of Jobs	**Time in milliseconds**	
	α=0.3, β=0.7	**α=0.7, β=0.3**
Job 1	150.2345	200.789
Job 2	220.65	100.567
Job 3	300.267	350.789
Job 4	430.567	234.78

In the above Table 3 the makespan for four jobs are calculated Compared to the previous Adaptive Scoring Job Scheduling Algorithm the makespan time is reduced after using the Enhanced Adaptive Scoring Job Scheduling Algorithm with replication strategy. Similarly the makespan can be calculated for 1000 jobs

Here the values α=0.3, β=0.7 represent the value for computational intensive jobs and α=0.7, β=0.3 represent the value of data intensive jobs. The makespan time of both the jobs are calculated.

6. CONCLUSIONS

The proposed Enhanced Adaptive Scoring Job Scheduling Algorithm with replication strategy method schedule jobs in dynamic heterogeneous grid environment. The algorithm divides the jobs into subtasks. The subtasks are replicated and the replicated job is assigned to the cluster. Jobs that are considered in this methodology are independent and the jobs are allocated to the cluster by computing cluster score. The job whether it is data intensive or computational intensive can also be identified without user specification and based on that the jobs can be scheduled.

ACKNOWLEDGEMENTS

I extend my gratitude to my supervisor, Dr.K.Kousalya M.E., Ph.D., for her valuable ideas and suggestions, which have been very helpful in the project

REFERENCES

[1]. A Somayeh., P Hossein., M Somayeh.,(2010), 'The impact of data replication on job scheduling performance in hierarchical data grid ',International journal on applications of graph theory in wireless ad hoc networks and sensor networks, Vol.2, No.3.

[2]. D. Paranhos, W. Cirne, F. Brasileiro, 'Trading cycles for information: using replication to schedule bag-to-tasks applications on computational grids',International Conference on Parallel and Distributed Computing (Euro-Par), Lecture Notes in Computer Science, vol. 2790, 2003, pp. 169–180.

[3]. H Yuan., X Qin., Li X., M HWu, (2005),'An improved ant algorithm for job scheduling in gird computing', in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, Vol. 5, 18–21, pp. 2957-2967.

[4]. Ruay-Shiung Chang , Chih-Yuan Lin, Chun-Fu Lin, 'An Adaptive Scoring Job Scheduling algorithm for grid computing', Information Sciences 207 (2012) 79–89 .

[5]. Ruay-Shiung Chang, Jih-Sheng Chang, Po-Sheng Lin,' An ant algorithm for balanced job scheduling in grids', Future Generation Computer Systems 25(2009) 20-27

[6] N M Shah ., M N B Zakaria., H Nazleein., A K B Mahmood., N Ken.,(2012), 'Design and Evaluation of Agent Based Prioritzed Dynamic Round Robin Scheduling algorithm for computational grids', AARI Procedia 1, pp. 531-543.

[7]. Ramya R, Shalini Thomas (2012), 'An Optimal Job Scheduling Algorithm in Computational Grids', International Conference on Communication, Computing and Information Technology

[8]. S Wang., I Hsu, Z I Huang., (2005), 'Dynamic scheduling methods for computational grid environment', International Conference on Parallel and Distributed Systems.

[9]. Wei L., Zhang X., Li Y., Li Y,(2012), 'An improved ant algorithm for grid task scheduling strategy', Physics Procedia 2, pp.1974-1981.

[10]. Li Y., Yang Y. and Zhou L. (2009), 'A hybrid load balancing strategy of sequential tasks for grid computing Environments', Future Generation Computer Systems, Vol.25, pp. 819-828.