An Effective Approach to Job Scheduling in Decentralized Grid Environment

G.K.Kamalam
Senior Lecturer/CSE
Kongu Engineering College
Perundurai, Erode

V.Murali Bhaskaran
Principal
Paavai College of Engineering
Pachal, Namakkal

ABSTRACT

Scheduling of jobs and resource management are the important challenging work in a grid environment. Processing time minimization of the jobs arriving at any computer site in a grid system is one of the major objectives in the research area of computing. In this paper, we propose a decentralized grid system model as a collection of clusters. We then introduce a decentralized job scheduling algorithms which performs intra cluster and inter cluster (grid) job scheduling. In this paper, we apply Divisible Load Theory (DLT) and Least Cost Method (LCM) to model the grid scheduling problem involving multiple resources within an intra cluster and inter cluster grid environment. The decentralized job scheduling algorithm is proposed for both reducing processing time and processing cost. The proposed decentralized hybrid job scheduling algorithm employs the DLT and LCM technique and decentralized divisible job scheduling algorithm employs DLT technique and produces reduced processing time and reduced processing cost. The result shows that the gap between the decentralized job scheduling algorithm and centralized job scheduling algorithm is widening as the number of jobs is increased.

Keywords

Job Scheduling, Heuristic Algorithm, Load Balancing, Cluster, Coordinator Node, Worker Node.

1. INTRODUCTION

A computational grid is emerging as a wide-scale distributed computing infrastructure where user jobs can be executed on either intra cluster or inter cluster computer systems [7,9]. Computational grids have the ability for solving large-scale scientific problems using heterogeneous and geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost and quality of service requirements [6,8]. Resource management and job scheduling in grid environment based on clusters is one of the challenging task [5]. Scheduling is an important problem in computational grid [14]. The grid environment is dynamic in nature, in other words, the number of resources and jobs to be scheduled are usually variable. This kind of feature of grid makes the scheduling problem a complex optimization problem [15]. The effective utilization of grid is the efficient scheduling of jobs to the available resources.

Many heuristic algorithms have been designed to solve the job scheduling problems. Based on the time at which the scheduling decisions are made, these algorithms are divided into static and dynamic scheduling [10]. In the static mode, a firm estimate of the completion time is made in advance of the actual execution. The advantage of the static model is that it is easier to program from the scheduler's point of view. The allocation of job to resource is fixed a priori, and estimating the cost of job is also easy. The disadvantage is that the cost estimate based on the static information is not accurate because in some situations, the resource selected for executing a job fails due to network failure, or is so heavily loaded with jobs that its response time is more than expected. Dynamic scheduling is applied when the jobs are arrived dynamically. It is used when it is difficult to estimate the computation time of a job [13].

The primary importance is to design an efficient scheduling algorithm for minimizing the total processing time of the jobs. This is achieved by distributing the jobs among the resources available in the grid system. The aim is to minimize the overall processing time and the overall processing cost using specified decentralized job scheduling algorithm. This scheduling technique is used to effectively schedule jobs to the available resources to reduce the processing time and processing cost of the jobs. This research is concerned with scheduling of jobs in distributed systems with divisible jobs.

The aim of this paper is to present a decentralized hybrid job scheduling algorithm and decentralized divisible job scheduling algorithm adapted to heterogeneous grid computing environment. In this paper, we compare two classes of scheduling algorithms centralized and decentralized job scheduling algorithms. We model the grid as a group of clusters. Group of users submit jobs to the various clusters. In centralized scheduling, the scheduler of each cluster is responsible for scheduling of submitted jobs. In decentralized scheduling, jobs though submitted locally can be migrated to another cluster in order to reduce the processing time of the jobs.

The rest of the paper is organized as follows: Section 2 summarizes the related work. Section 3 discusses the decentralized grid architecture model. Section 4 describes the design of decentralized job scheduling algorithm. Section 5 compares the decentralized and centralized job scheduling algorithms. Section 6 concludes the paper.

2. RELATED WORK

Divisible load theory is designed to solve the challenging problem of allocating the independent jobs to the available resources on a grid [2, 3, 12]. A divisible job is one that can be arbitrarily divided into independent sub jobs. These sub jobs can be executed in any order on platforms ranging from one single cluster to large distributed grid systems [11]. The processing time of each sub job is small compared to the time to process the whole job.

In LCM method, the jobs are allocated to the resource with the least allocation cost [1].

In [1] introduced a decentralized load balancing algorithm, that considers load index as a decision parameter for scheduling of jobs within intra cluster and inter cluster grid environment.

3. DECENTRALIZED GRID ARCHITECTURE MODEL

We have proposed a decentralized grid scheduling architecture model for scheduling jobs in grid consists of a collection of clusters where cluster server are treated as Coordinator Nodes (CN). Each cluster consists of more number of Worker Nodes (WN). Each worker node has different processing powers. The worker nodes within the cluster are interconnected through a high speed local area network. The clusters are connected through a wide area network.

The jobs generated by the users are submitted to the coordinator nodes. A scheduler then partitions the jobs into sub jobs and schedules. Dispatcher then dispatches the partitioned sub jobs to the appropriate resources in the intra cluster or inter cluster grid environment. Grid Information Centre (GIC) maintains the memory and CPU utilization value along with the registration information of CN.

4. DECENTRALIZED JOB SCHEDULING ALGORITHMS (DJSA)

4.1 Decentralized Makespan Local Job Scheduling Algorithm (DMLJSA)

The grid system consists of a cluster with a Coordinator Node denote CN, connected to a set of n Worker Node denoted as WN_1 ,WN_2,…,WN_n. Each user is assigned to a particular cluster. The jobs submitted by the user will be executed in the originating cluster or it will be migrated to any of the clusters.

Let k be the number of jobs and n be the number of worker nodes in each cluster. Let l represent the number of users and m represents the number of clusters in a grid system. More formally,

Set of jobs, $J = \{J1, J2, J3, \dots, Jk\}$

Set of clusters, $C = \{C1, C2, C3, \dots, Cm\}$

Set of users , $U = \{U1, U2, U3, \dots, Ul\}$

Set of workernodes,
$WN = \{WN11, \dots. W1n, W21, \dots. W2n, \dots. WNm1, WNm2, \dots WNmn\}$

where

$k >= 1;\ n > 1;\ l >.= 1;\ m > 1;$

WN_{nm} is the mth worker node of nth cluster.

J_k is the kth job of the set of jobs submitted to the grid system.

PC_{nm} is the processing cost of mth worker node of nth cluster.

CT_{ij} is the completion time of jth worker node of ith cluster.

When the job enters cluster, the scheduler receives the information from GIC and finds the worker node from the entire cluster whose completion time is minimum and assigns the job to that particular worker node.

Decentralized makespan local job scheduling algorithm is as follows:

Step1: The completion time information of each WN of each cluster is updated at GIC or CN of a cluster periodically.

Step2: If the job set J is empty then go to step6.

Step3: If a new job arrives at CN of a cluster C_i, the scheduler receives the completion time information of a worker node of each cluster from GIC and assigns the job to the worker node with minimum processing time. The processing time is calculated as follows:

```
        $makespan = clusterlist[0].WNList[0].uptime$
$for\ each\ cluster\ Ci, 1 \leq\ i \leq noofclusters$
        $for\ each\ worker\ node\ WNj, 1 \leq j \leq workernodes$
$if\ (clusterlist[i].WNList[j].uptime > makespan)$
$makespan = clusterlist[i].WNList[j].uptime$
```

Step4: Repeat step 4 until all jobs is scheduled.

Step5: The processing cost is calculated as:

```
        $for\ i = 1 to\ m$
        $for\ j = 1\ to\ n$
            $Total\ cost = Total\ cost + CTij * PCij$
```

Step6: END.

4.2 Decentralized Cost Local Job Scheduling Algorithm (DCLJSA)

When the job enters cluster, the scheduler receives the information from GIC and finds the worker node from the entire cluster whose completion time and processing cost is minimum and assigns the job to that particular worker node.

Decentralized cost local job scheduling algorithm is as follows:

Step1: The completion time information of each WN of each cluster is updated at GIC or CN of a cluster periodically.

Step2: If the job set J is empty then go to step6.

Step3: If a new job arrives at CN of a cluster C_i, the scheduler receives the completion time information of a worker node of each cluster from GIC and assigns the job to the worker node with minimum processing time and minimum processing cost. The processing time is calculated as follows:

```
        $makespan = clusterlist[0].WNList[0].uptime$
$for\ each\ cluster\ Ci, 1 \leq\ i \leq noofclusters$
        $for\ each\ worker\ node\ WNj, 1 \leq j \leq workernodes$
$if\ (clusterlist[i].WNList[j].uptime > makespan)$
$makespan = clusterlist[i].WNList[j].uptime$
```

Step4: Repeat step 4 until all jobs is scheduled.

Step5: The processing cost is calculated as:

```
    $for\ i = 1 to\ m$
    $for\ j = 1\ to\ n$
        $Total\ cost\ = Total\ cost\ + CTij * PCij$
```

Step6: END.

4.3 Decentralized Divisible Job Scheduling Algorithm (DDJSA)

A divisible job J_i is divided into jobs of equal size.

Let k be the number of jobs and q is the number of partitions of a job.

$J = \{J1, J2, J3, \dots, Jk\}$

where J is a set of k jobs and k>=1

Each job is split into q sub jobs

$Ji = \{SJi1, SJi2, \dots. SJiq\}$

where q>=1

Decentralized Divisible job scheduling algorithm is:

Step1: The completion time of WN of every cluster is maintained at GIC or CN of cluster periodically.

Step2: If J is empty, go to step 8.

Step3: If a job arrives, divide the job into sub jobs to the maximum of five partitions.

Step4: Find the worker node with minimum processing time from all the clusters as follows:

```
    $makespan = clusterlist[0].WNList[0].uptime$
$for\ each\ cluster\ Ci, 1 \le\ i \le noofclusters$
        $for\ each\ worker\ node\ WNj, 1 \le j \le workernodes$
$if\ (clusterlist[i].WNList[j].uptime\ >\ makespan)$
$makespan = \ clusterlist[i].WNList[j].uptime$
```

Step5: Repeat step 4 until all the 'q' sub jobs are processed.

Step6: Repeat step 3 to 4 until all the jobs are scheduled.

Step7: Calculate the processing cost.

```
  $for\ i = 1\ to\ m$
      $for\ j = 1\ to\ n$
          $Total\ cost\ = Total\ cost\ + CTij * PCij$
```

Step 8: END.

4.4 Decentralized Hybrid Job Scheduling Algorithm (DHJSA)

In static scheduling the scheduler needs to know the execution time of every job in advance. If this information is not accurate the scheduling decisions may be inefficient. In real time systems, the estimation of job execution time is a hard problem. To overcome this inefficiency, we propose the decentralized hybrid job scheduling algorithm. First, the users are assigned to the various clusters. Jobs that are mutually independent are submitted to the different CN of various clusters. The scheduler than partitions the job into sub jobs to the maximum of five partitions and are placed in a global job set J. The CN receives the completion time information of each worker node in each cluster and communicates this information to the GIC periodically, which is later on used for the allocation of sub jobs to the worker node among clusters. Upon receiving this information from GIC, the scheduler then schedules the sub jobs to the worker node of any cluster whose processing time * processing cost is minimum.

Decentralized hybrid job scheduling algorithm is described as:

Step1: If there is any completion time information from CN or WN then update the information at GIC or CN.

Step2: If J is empty then go to step 10.

Step 3: If a job J_i completes the execution of all its sub jobs and was migrated to another cluster then dispatch this job along with results to the generated cluster and remove the job from the job set J.

Step 4: If a new job arrives at CN of any cluster C_i then partition the job into maximum of 5 equal partitions and then add it to the job set.

Step 5: Among all the clusters find the worker node with minimum processing time and allocation cost. The processing time is

```
$makespan = clusterlist[0].WNList[0].uptime$
$for\ each\ cluster\ Ci, 1 \le\ i \le noofclusters$
        $for\ each\ worker\ node\ WNj, 1 \le j \le workernodes$
$if\ (clusterlist[i].WNList[j].uptime\ >\ makespan)$
$makespan = \ clusterlist[i].WNList[j].uptime$
```

Step6: CN at cluster C_i then dispatches the sub job to the worker node WN_{min}

Step7: Repeat step 5 to 6 until all sub jobs is scheduled.

Step8: Repeat step 4 to 6 until the job set is empty.

Step9: Calculate the processing cost.

```
  $for\ i = 1\ to\ m$
      $for\ j = 1\ to\ n$
          $Total\ cost\ = Total\ cost\ + CTij * PCij$
```

Step10: END.

5. COMPUTATIONAL RESULTS

In this section, the performance of decentralized job scheduling algorithm is studied based on the simulation parameters of [4] and are shown in Table 1.

Table 1. Simulation Parameters

Parameters	Value
No. of Clusters	10
No. of worker nodes per cluster	10
Processing power of worker	500– 5000 MIPS
Job length	2,50,000-6,50,000 MI
Cost	1-5 G$ unit
No. of users	5
No. of Jobs	50 - 500

The DJSA performance measurement is measured based on the three parameters: Total processing time, Total cost and Number of jobs. The result of DJSA and without DJSA (WDJSA) shows that the DJSA is more efficient. The total processing time of completing the jobs of DDJSA algorithm is minimum compared to that of the WDJSA algorithm is shown in Table 2 and Table 3 respectively. The total processing cost of completing the jobs of DHJSA algorithm is minimum compared to that of the WDJSA algorithm is shown in Table 4 and Table 5 respectively. Figure 2 and Figure 3 shows the impact of DJSA and WDJSA on the total processing time and the total cost of completing the jobs.

Table 2. Total Processing Time of WDJSA Algorithm

No. of Jobs	WDCLJSA	WDMLJSA	WDHJSA	WDDJSA
50	907	502	486	313
100	2811	1508	2333	1102
150	4644	1829	2816	1116
200	5863	2546	3037	1324
250	6300	3340	3973	2114
300	7362	3432	3651	1710
350	10695	4581	7173	3083
400	16615	6168	10272	3844
450	5665	4519	3578	2866
500	15177	5618	10179	3803

Table 3. Total Processing Time of DJSA Algorithm

No. of Jobs	DCLJSA	DMLJSA	DHJSA	DDJSA
50	222	140	174	120
100	532	262	352	209
150	473	290	307	232
200	827	356	451	232
250	793	431	524	298
300	899	433	540	255
350	1328	564	973	424
400	2028	754	1354	518
450	679	546	462	390
500	1336	677	939	475

Table 4. Total Processing Cost of W DJSA Algorithm

No. of Jobs	WDMLJSA	WDCLJSA	WDDJSA	WDHJSA
50	10732	7676	5534	4036
100	30390	28942	23745	22855
150	49015	46261	29796	28363
200	70534	55112	35635	27608
250	91251	58829	59209	38084
300	89777	69827	44456	34643
350	125437	104897	83984	70545
400	186485	163821	114402	99863
450	124744	111332	77631	68897
500	177196	148306	117963	99139

Table 4. Total Processing Cost of DJSA Algorithm

No. of Jobs	DMLJSA	DCLJSA	DDJSA	DHJSA
50	7686	4506	3735	2324
100	25648	21594	20625	17903
150	42896	39250	25703	22725
200	62824	40799	30950	20627
250	75214	41866	49006	27097
300	83453	60441	40668	26358
350	118321	96057	80953	65301
400	179621	153078	109910	92300
450	120180	104815	74257	65101
500	149507	103360	100485	69278

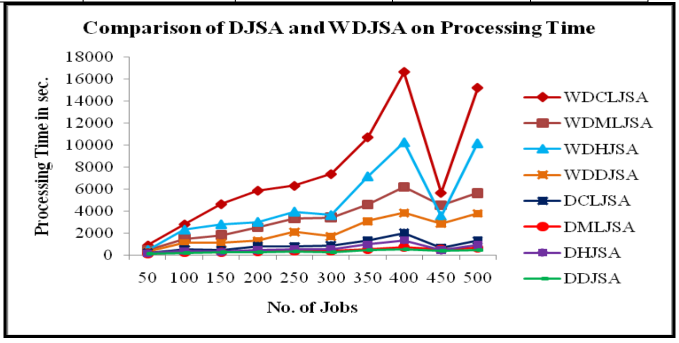

Fig 1: Impact of DJSA and WDJSA on Processing Time

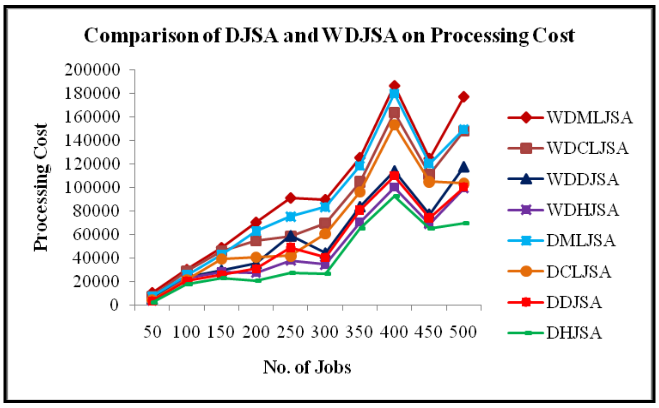

Fig 2: Impact of DJSA and WDJSA on Total Cost

6. CONCLUSION

In this paper, we present an effective decentralized job scheduling algorithm for grid environment. The result shows that this algorithm reduces the total processing time and the total cost and finally, the resource utilization is more and the load is balanced across the grid environment. As to future work, is to implement our DJSA algorithm in an actual grid environment.

7. REFERENCES

[1] Syed Nasir Mehmood Shah, Ahmad Kamil Bin Mahmood, and Alan Oxley 2010, "Hybrid Resource Allocation for Grid Computing", in Proceedings of the IEEE Second International Conference on Computer Research and Development, 426 – 431.

[2] D.Yu, T.G.Robertazzi 2003, "Divisible Load Scheduling for Grid Computing", in Proceedings of the International Conference on Parallel and Distributed Computing Systems.

[3] Murugesan, and C.Chellappan 2009, "An Economical Model for Optimal Distribution of Loads for Grid Applications", in Internal Joural of Computer and Network Security, Vol.1, No.1.

[4] P.K.Suri, and Manpreet Singh 2010, "An Efficient Decentralized Load Balancing Algorithm for Grid", in Proceeding ofStar the IEEE Second International Advance Computing Conference,10 – 13.

[5] M.Balajee, B.Suresh, M.Suneetha, V.Vasudha Rani, and G.Veerraju 2010,"Preemptive Job Scheduling with Priorities and Starvation cum Congestion Avoidance in Clusters ", in Proceedings of the IEEE Second International Conference on Machine Learning and Computing , 225 - 229.

[6] N.Malarvizhi, and Dr.V.Rhymend Uthaiaraj 2009, "A Minimum Time to Release Job Scheduling Algorithm in Computational Grid Environment", in Proceedings of the 5th IEEE Joint International Conference on INC, IMS, and IDC, 13 - 18.

[7] I.Foster, and C.Kesselman 1999,"The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers,USA.

[8] M.Baker, R.Buyya, and D.Laforza 2002,"Grids and Grid Technologies for Wide-area Distributed Computing", Software-Practice and Experience, Vol.32, No.15, 1437 – 1466.

[9] I.Foster, C.Kesselman, and S.Tuecke 2001,"The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal on High Performance Computing Applications, Vol.15, No.3, 200 – 222.

[10] G.Manimaran, and C.Siva Ram Murthy 1998,"An Efficient Dynamic Scheduling Algorithm for Multiprocessor Real-time Systems", IEEE Transactions on Parallel and Distributed Systems, Vol.9, No.3, 312 - 319.

[11] N.Amano, J.O.Gama, and F.Silva 2003, "Exploiting Parallelism in Decision Tree Induction', in Proceedings from the ECML/PKDD Workshop on Parallel and Distributed Computing for Machine Learning, 13 – 22.

[12] V.Bharadwaj, D.Ghose, and T.G.Robertazzi 2008,"Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems", Cluster Computing, Vol.6, No.1, 7 – 17.

[13] M.Chetper, F.Claeys, B.Dhoedt, F.De Turck, P.Vanrollegham, and P.Demeester 2006,"Dynamic Scheduling of Computationally Intensive Applications on Unreliable Infrastructures", in Proceedings of the Second European Modeling and Simulation Symp.

[14] C.Grimme, J.Lepping, A.Papaspyrou, P.Wieder, R.Yahyapour, A.Oleksiak, O.Waldrich, and W.Ziegler 2007, "Towards a Standards-based Grid Scheduling Architecture", CoreGRID Technical Report TR-0123, Institute on Resource Management and Scheduling.

[15] K.Kurowski, J.Nabrzski, A.Oleksiak, and J.Weglarz 2006, "Scheduling Jobs on the Grid-Multicriteria Approach", Computational Methods in Science and Technology, Vol.12, No.2, 123 – 138.