American Journal of Applied Sciences 9 (8): 1294-1299, 2012
ISSN 1546-9239

Novel Adaptive Job Scheduling Algorithm on Heterogeneous Grid Resources

[1]G.K. Kamalam and [2]V. Murali Bhaskaran
[1]Department of CSE, Kongu Engineering College, Perundurai, Erode, India
[2]Pavaai College of Engineering, Pachal, Namakkal, India

Abstract: Grid provides an infrastructure for sharing geographically distributed heterogeneous resources to process many applications and mainly used for solving scientific problems that requires more computation time. **Problem statement:** Grid is a dynamic environment, where the resources may join or leave the environment at any time and the jobs also arrives at different intervals of time. To meet the demands and requirements of the dynamic environment, to maximize the resource utilization and to minimize the makespan an effective grid scheduling technique is needed. **Approach:** We propose grid architecture as a collection of clusters with multiple worker nodes in each cluster. We propose a new scheduling algorithm Novel Adaptive Decentralized Job Scheduling Algorithm (NADJSA) that applies both Divisible Load Theory (DLT) and Least Cost Method (LCM) and also considers the user demands. **Results:** The proposed Novel Adaptive Decentralized Job Scheduling Algorithm is compared with the Decentralized Hybrid Job Scheduling Algorithm. **Conclusion:** The proposed Novel Adaptive Decentralized Job Scheduling Algorithm minimizes the makespan, improves the resource utilization and satisfies the user demands and well suits for the grid environment.

Key words: Grid scheduling, cluster, coordinator node, worker node, heterogeneous resources

INTRODUCTION

Grid system consists of geographically distributed heterogeneous resources that belong to various administrative domains. The dynamics and heterogeneity nature of the grid environment makes the scheduling problem a challenging one. In general, job scheduling in heterogeneous grid environment is an NP-hard problem (Manavalasundaram and Duraiswamy, 2012).

Divisible loads are classified as arbitrarily divisible loads and modularly divisible loads. Modularly divisible loads are divided into pre-defined modules. Between the modules, precedence relations may exist. Arbitrarily divisible loads are divided into partitions of arbitrary lengths. No precedence relations exist between the loads and these arbitrarily divisible independent processing loads can be processed on more than one processor (Shokripour and Othman, 2009a).

Divisible Load Theory (DLT) provides time-optimal processing of jobs. The ten importance of DLT are: (1) A tractable model (2) Interconnection topologies (3) Equivalent networks (4) Installments and sequencing (5) Scalability (6) Metacomputing accounting (7) Time-varying modelling (8) Unknown system parameters (9) Extending realism (10) Experimental results (Robertazzi, 2003).

Least Cost Method (LCM) allocates the job to the available resources in the grid with minimum processing cost.

The aim of the scheduling algorithm is to minimize the processing time of the job. Optimizing processing time of the job is done by dividing the jobs into sub jobs and allocating the sub jobs to the worker node of different clusters in a decentralized grid environment.

The proposed Novel Adaptive Decentralized Job Scheduling Algorithm employs the DLT and LCM and allocates the job efficiently to the available resources in the decentralized grid with minimum makespan and minimum processing cost.

MATERIALS AND METHODS

Related research: The powerful tool for efficient scheduling of computing loads is the divisible load theory. It is especially emerged for scheduling of parallel loads that are divisible among the processors and links. Divisible load theory is a linear mathematical model, the computing loads can be partitioned arbitrarily and can be executed in any order and it provides optimal processing of the computational loads (Shokripour and Othman, 2009b).

In LCM, the arbitrary divisible independent processing loads are allocated to the resource with the least allocation cost.

Corresponding Author: Kamalam, G.K., Department of CSE, Kongu Engineering College, Perundurai, Erode, India

Shah *et al*. (2010a), the job is divided into tasks of equal size and the task is allocated to the processor with the least allocation cost. If more than one processor has the same least allocation cost, then the processor with the maximum available processing unit is selected. The task to be scheduled on this processor is selected based on the processing time of the task. The task which has the maximum processing time is selected and scheduled on the processor.

Shah *et al*. (2010b), the processor and the job with the least allocation cost are selected. If more than one processor has the same least allocation cost, then select the next least allocation cost for both the processor and the job. Among the processor that has the least cost allocation select the more processing unit available processor and select the job with maximum workload and allocate to that processor. Select the next least allocation cost for the minimum job workload.

Decentralized grid model: We have proposed a decentralized grid model as a collection of clusters where cluster servers are treated as Coordinator Nodes (CN) denoted as CN = {CN1, CN2, ..., CNm}. Each cluster consists of multiple Worker Nodes (WN) denoted as WN = {WNi1, WNi2, ..., WNin}. Each worker node has different processing powers. The grid environment is dynamic; the worker nodes can leave or join the grid at any time. Each worker node has its own availability time, the time at which the worker node is available in the grid. The worker nodes within the cluster are interconnected through a local area network. The clusters are connected through a wide area network. Grid Information Centre (GIC) maintains the CPU and memory utilization value of all the nodes in the gird. Coordinator nodes of each cluster provide this information to GIC periodically (Suri and Singh, 2010).

Each user Ui owns a cluster Ci. The set of users U is denoted as U = {U1, U2, ..., Uo}. The set of all jobs submitted by the user U of a cluster C is denoted by J. The set of jobs is denoted as J = {J1, J2, ..., Jk}. Each job is split into sub jobs as Ji = {SJi1, SJi2, ..., SJil}.

In a decentralized dynamic grid environment the scheduling of jobs is a linear programming transportation problem. An efficient novel approach is essential for scheduling of jobs originating from any cluster to any other cluster at minimum transportation cost. Scheduling also considers the various parameters like minimum makespan, minimum processing cost, availability time of the worker node, deadline of the job, transportation cost, the communication time to transfer the job submitted in one cluster to the other cluster for processing.

Existing research: A divisible job Ji is divided into sub jobs to the maximum of five partitions. Let k be the number of jobs and q be the number of partitions of a job (Kamalam and Bhaskaran, 2011a, 2011b):

J = {J1, J2,..., Jk}
Ji = {SJi1, SJi2,..., SJiq}
Where k >= 1 and q >= 1

The user submits the job to the coordinator node. The coordinator node contacts GIC and gets the information of memory and CPU utilization of each worker node in the grid and allocates the sub job to the node with the minimum processing time and processing cost.

Decentralized Hybrid Job Scheduling Algorithm (DHJSA):

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 9.
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 J.
Step 5: Among all the clusters find the worker node with minimum processing time and allocation cost. The processing time is:

```
for i = 1 to m
for j = 1 to n
WN min = min {(CTij+
(job length/job count/
Cluster [i]. WN [j]. processing power))*
Cluster [i]. WN [j]. allocation cost}
```

Step6: CN at cluster C_i then dispatches the sub job to the worker node WN_{min}
Step7: Repeat step 5-6 until all sub jobs is scheduled.
Step8: Repeat step 4-7 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 +
    (Completion Time (WNij)*
```

processing Cost (WNij))

Step10: END.

Novel Adaptive Decentralized Job Scheduling Algorithm (NADJSA): When the user submits the job, the job is arbitrarily partitioned into sub jobs to the maximum of five partitions. The CN receives the information from the GIC and selects the worker node for scheduling among the entire cluster considering the following parameters: Processing cost, Processing time, Transportation cost, Availability time of the worker node and Deadline of the job.

Novel Adaptive Decentralized Job Scheduling Algorithm (NADJSA) is as follows:

Step 1: If there is any completion time information from CN or WN then update the information at GIC or CN.
Step 2: If J is empty then go to step 12.
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: The initial processing time is calculated as:

```
minval = Cluster [0]. allocation cost* (job length/
Cluster [0]. WN [0]. processing power) +
(Cluster [0]. WN [0]. allocation cost*
Cluster [0]. WN [0]. processing time)
```

Step 5: 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 6: Among the entire clusters find the worker node with minimum processing time, transfer time and allocation cost. Also check the availability of the worker node for allocating the sub job to the particular worker node. The processing time is:

```
for i = 1 to m
    for j = 1 to n
        if (Cluster [i]. WN [J]. allocation cost*
            ((job length/job count)/
            Cluster [i]. WN [j]. processing power) +
            Cluster [i]. WN [j]. allocation cost*
            Cluster [i]. WN [j]. processing time))
            < minval
    if ((availability [i] [j]) >
        (Cluster [i] WN [j]. processing time +
        ((transfer time[Clusterorigin] [i]) +
        ((job length/job count)/
            Cluster [i] WN [j]. processing power)))
        Cluster [min C]. WN [min WN]. processing time + =
        ((job length/job count) /
        Cluster [min C]. WN [min WN]. processing
power)
        + transfer time [Cluster origin] [min C]
```

Step 7: The total processing time of a job is calculated as:

```
Total processing time + =
((job length/job count)/
Cluster [min C]. WN [min WN]. processing power))
+ transfer time [Cluster origin] [min C]
```

Step 8: CN at cluster C_i then dispatches the sub job to the worker node WN_{min}
Step 9: Repeat step 6-8 until all sub jobs is scheduled.
Step 10: If total processing time of the job is less than the deadline of the job

```
        hit count = hit count + 1
else
        miss count = miss count + 1
```

Step11: Repeat step 5 to 10 until the job set is empty.
Step12: Calculate the processing cost.

```
for i = 1 to m
for j = 1 to n
Total cost = Total cost +
(Completion Time (WNij)*
processing Cost (WNij))
```

Step13: END.

RESULTS

We compare the analysis of our proposed Novel Adaptive Decentralized Job Scheduling Algorithm with the existing Decentralized Hybrid Job Scheduling Algorithm based on the simulation parameters of (Suri and Singh, 2010) and are listed in Table 1.

Table 1: Simulation parameters

Parameters	Value
No. of clusters	10
No. of worker nodes per cluster	10
Processing power of worker node	500-5000 MIPS
Job length	2, 50, 000-6, 50, 000 MI
Cost	1-3 G$ unit
No. of users	5
No. of jobs	100-1000
Deadline of the job	600-650 sec
Bandwidth	1000-1500Mb/s
Available time of the worker nodes	400-500 sec

DISCUSSION

The performance of the job scheduling algorithm is based on the three parameters: Total processing time, Total processing cost and Number of jobs. The performance is compared by varying the number of jobs.

Table 2 and 3 show the processing time and processing cost obtained by the Decentralized Hybrid Job Scheduling Algorithm and the proposed Novel Adaptive Decentralized Job Scheduling Algorithm.

Graphical representation of Table 2 in Fig. 1 shows that the proposed Novel Adaptive Decentralized Job Scheduling Algorithm provides a better makespan than the Decentralized Hybrid Job Scheduling Algorithm.

Graphical representation of Table 3 in Fig. 2 shows that the proposed Novel Adaptive Decentralized Job Scheduling Algorithm provides a minimum processing cost than the Decentralized Hybrid Job Scheduling Algorithm.

Table 2: Processing time

No. of Jobs	NADJSA	DHJSA
100	9699	13864
200	56098	85853
300	125053	174217
400	170644	215192
500	257405	333918
600	323130	384343
700	427995	528067
800	518738	600584
900	655417	756436
1000	616090	695965

Table 3: Processing cost

No. of Jobs	NADJSA	DHJSA
100	39234	130577
200	161087	307107
300	194270	412466
400	425361	718130
500	622570	966938
600	435300	828803
700	965827	1436470
800	1153341	1786471
900	756884	1228351
1000	1341325	1960096

Table 4: Comparison based on miss count

No. of Jobs	NADJSA	DHJSA
100	16	57
200	47	81
300	103	134
400	143	213
500	206	262
600	273	330
700	293	359
800	394	459
900	415	499
1000	434	557

Table 5: Comparison based on hit count

No. of Jobs	NADJSA	DHJSA
100	84	43
200	153	119
300	197	166
400	257	187
500	294	238
600	327	270
700	407	341
800	406	341
900	485	401
1000	566	443

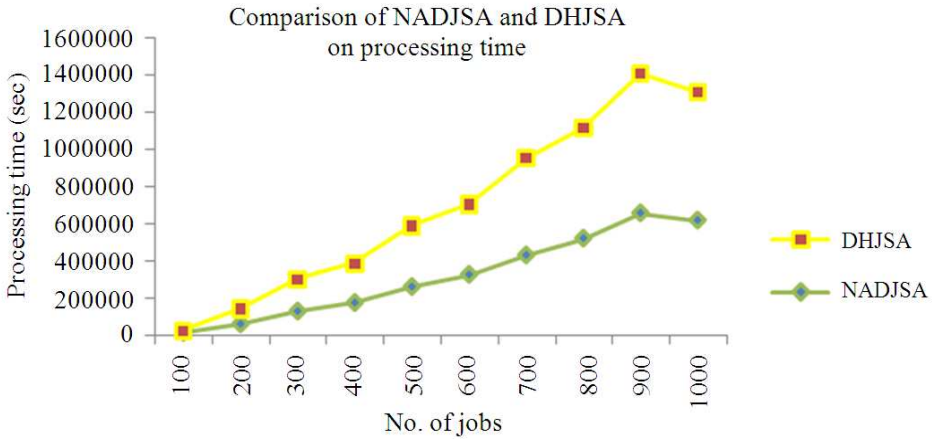

Fig. 1: Impact of NADJSA and DHJSA on Processing Time

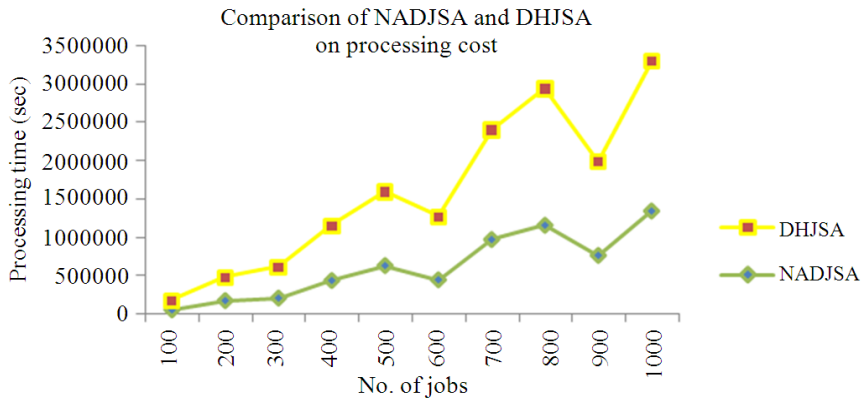

Fig. 2: Impact of NADJSA and DHJSA on Processing Cost

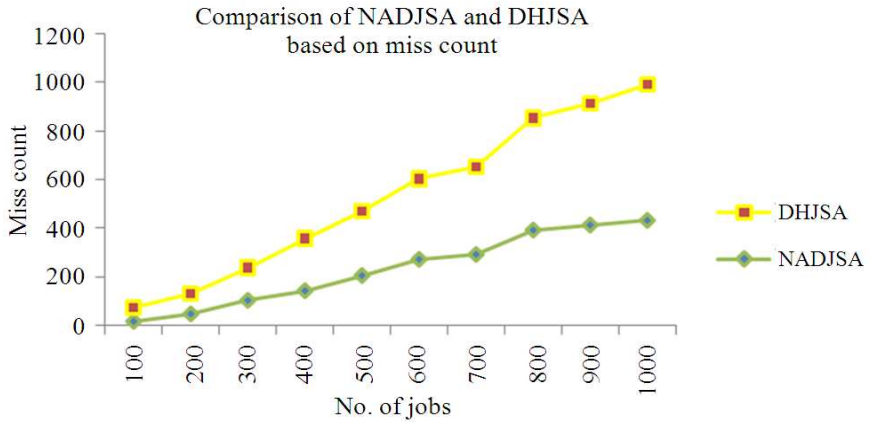

Fig. 3: Impact of NADJSA and DHJSA on Miss Count

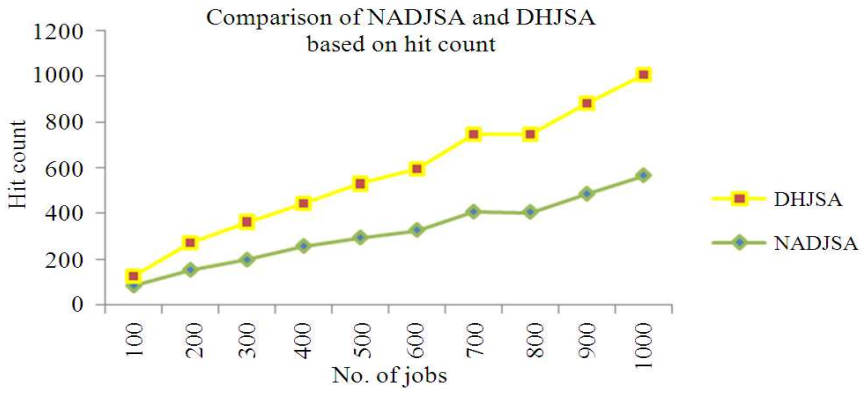

Fig. 4: Impact of NADJSA and DHJSA on Hit Count

The proposed Novel Adaptive Decentralized Job Scheduling Algorithm allocates the job to the worker node based on the deadline of the job. Table 4 and Table 5 shows the comparison based on the miss and hit count. Miss represents the number of jobs that are completed after the user deadline. Hit represents the successful completion of jobs within the user deadline demand.

Graphical representation of miss and hit count is presented in Fig. 3 and 4. The simulation result shows that the proposed Novel Adaptive Decentralized Job Scheduling Algorithm shows a

high hit rate and less miss rate than the Decentralized Hybrid Job Scheduling Algorithm.

CONCLUSION

In this study, we presented an efficient Novel Adaptive Decentralized Job Scheduling Algorithm for a decentralized grid environment. The proposed Novel Adaptive Decentralized Job Scheduling Algorithm aims at minimum cost (Processing time, Processing cost, Transfer cost). The Novel Adaptive Decentralized Job Scheduling Algorithm achieves minimum makespan and minimum processing cost than the Decentralized Hybrid Job Scheduling Algorithm. The result shows that the proposed Novel Adaptive Decentralized Job Scheduling Algorithm reduces the makespan and processing cost, satisfies the user demand, improves the resource utilization and balances the load across the grid environment.

REFERENCES

Shokripour, A. and M. Othman, 2009a. Survey on divisible load theory. Proceedings of the Spring Conference International Association of Computer Science and Information Technology, Apr. 17-20, IEEE Xplore Press, Singapore, pp: 9-13. DOI: 10.1109/IACSIT-SC.2009.55

Shokripour, A. and M. Othman, 2009b. Survey on divisible load theory and its applications. Proceedings of the International Conference on Information Management and Engineering, Apr. 3-5, IEEE Xplore Press, Kuala Lumpur, pp: 300-304. DOI: 10.1109/ICIME.2009.59

Kamalam, G.K. and V.M. Bhaskaran, 2011a. An effective approach to job scheduling in decentralized grid environment. Int. J. Comput. Appli., 24: 26-30. DOI: 10.5120/2914-3838

Kamalam, G.K. and V.M. Bhaskaran 2011b. An efficient hybrid job scheduling algorithm for computational grids. Int. J. Comput. Appli.

Manavalasundaram, V.K. and K. Duraiswamy, 2012. Integrated resource and cost management scheme for computational grids. J. Comput. Sci., 8: 538-544. DOI: 10.3844/jcssp.2012.538.544

Suri, P.K. and M. Singh, 2010. An efficient decentralized load balancing algorithm for grid. Proceedings of the IEEE 2nd International Advance Computing Conference, Feb. 19-20, IEEE Xplore Press, Patiala, pp: 10-13. DOI: 10.1109/IADCC.2010.5423048

Shah, S.N.M., A.K.B. Mahmood and A. Oxley, 2010a. Modified least cost method for grid resource allocation. Proceedings of the 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Oct. 10-12, IEEE Xplore Press, Huangshan, pp: 218-225. DOI: 10.1109/CyberC.2010.47

Shah, S.N.M., A.K.B. Mahmood and A. Oxley, 2010b. Hybrid resource allocation method for grid computing. Proceedings of the 2nd International Conference on Computer Research and Development, May 7-10, IEEE Xplore Press, Kuala Lumpur, pp: 426-431. DOI: 10.1109/ICCRD.2010.86

Robertazzi, T.G., 2003. Ten reasons to use divisible load theory. Computer, 36: 63-68. DOI: 10.1109/MC.2003.1198238