AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Best Cluster Decentralized Job Scheduling Algorithm for Scheduling Jobs on Heterogeneous Grid Environment

Dr.G.K. Kamalam,

Department of Information Technology, Kongu Engineering College, Perundurai. Erode, Tamilnadu, India.

ARTICLE INFO

Article history:
Received 10 October 2014
Received in revised form
22 November 2014
Accepted 28 November 2014
Available online 1 December 2014

Keywords:
Cluster, Divisible Load Theory, Job Scheduling, Coordinator Node, Worker Node

ABSTRACT

Background: Grid Computing provides highly scalable and utmost high performance mechanisms for discovering and negotiating access to the computing resources among infinite number of geographically distributed groups, to solve complex scientific or technical problems. Current problems in science and engineering are more complicated and require more computing power to analyze and solve. Grid computing aims to integrate virtualized, enabling the sharing, managing and aggregation of idle resources on the Internet to facilitate utilization. **Objective:** to solve complicated computing tasks and an efficient scheduling algorithm is essential for the successful execution of the grid applications. This paper addresses the performance metrics of scheduling complicated tasks by defining strategies to reduce the makespan, reduce the idle time of the resources and thereby improves the resource utilization. **Results:** Experimental results illustrate that the performance of the proposed strategy Best Cluster Decentralized Job Scheduling Algorithm (BCDJSA) give good results. **Conclusion:** The proposed algorithm BCDJSA minimizes the makespan and well suits for the grid environment.

© 2014 AENSI Publisher All rights reserved.

To Cite This Article: Dr.G.K. Kamalam., Best Cluster Decentralized Job Scheduling Algorithm for Scheduling Jobs on Heterogeneous Grid Environment. ***Aust. J. Basic & Appl. Sci.,*** *8(18): 171-177, 2014*

INTRODUCTION

Grid is a dynamically shared environment. It comprises of geographically distributed heterogeneous computational resources of multi-institutional domains with different usage policies (Foster and Kesselman, 1999). The idle resources of multi-institutional domains form a virtual organization (Caron *et al*., 2007).The virtual organization provides coordinated resource sharing to solve complex scientific problems that require enormous computing power (Chang *et al*., 2009). Grid Computing provides huge contributions to scientific research. It provides the data and resources that are geographically distributed around the world helping scientists to analyze and store massive collection of data (Baker *et al*., 2012). Jobs submitted in the grid environment are heterogeneous in nature. Jobs may be computing-intensive or data-intensive in nature. The computing-intensive job needs lot of computing power for completion and the data-intensive jobs need lot of bandwidth to transmit the data files across the geographically distributed environment for processing (Tierney *et al*., 2000). To achieve high performance, the Grid computing identifies the geographically distributed resources, selects the appropriate resources for executing the job to the selected resources (Wei *et al*., 2010). To efficiently utilize the power of Grid computing, an efficient job scheduling algorithm is essential. Job Scheduling is an NP-Complete problem (Garey and Johnson, 1979). To schedule the jobs in a distributed heterogeneous grid environment efficiently is a new challenge. This paper proposes a new scheduling algorithm Best Cluster Decentralized Job Scheduling Algorithm to reduce the completion time of the job.

The objective of job scheduling is to find a feasible schedule that minimizes the scheduling criteria, the makespan (Xiao, 2007). Makespan is the overall completion time of all the jobs submitted in the grid (Venugopal *et al*., 2006).

Over the past decade, a new linear mathematical tool created to improve the performance analysis of the systems, which include communication and computation problems in parallel and distributed environment. The key feature of Divisible Load Theory (DLT) is that, it uses the linear mathematical model and partitions arbitrarily the communication and computation loads into fractions and schedules the loads among the number of numerous processors and links.

Corresponding Author: Dr.G.K.Kamalam, Department of Information Technology, Kongu Engineering College, Perundurai, Erode, Tamilnadu. India.
Ph: +91 9578092100, E-mail: kamalamparames@gmail.com

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 171-177

DLT applies for large data intensive computational problems in grid computing, modeling and scheduling of some meta-computing applications, parallel and distributed computing problem, and architectural issue in parallel and distributed computing environment. DLT is using for scheduling the kind of divisible tasks such as processing of massive experimental data, image processing applications, video processing applications, network applications, mathematical computation, biological application, database and data grid, and sensor network.

Literature Review:

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).

In Caron *et al*. (2007), the authors proposed a hybrid system called DIRAC (Distributed Infrastructure with Remote Agent Control). This hybrid system develops a model of many clusters of heterogeneous nodes. Decentralized adaptive and opportunistic approaches perform well in comparison to a centralized approach.

In (Shah *et al*., 2010 a), the algorithm divides the job into tasks of equal size and allocates the task to the processor with the least allocation cost. If more than one processor has the same least allocation cost, than the algorithm selects the processor with the maximum available processing unit. The algortihm reduces the processor availability time and the job workload. The process repeats until the processor availability, or the job workload becomes zero.

In (Shah *et al*., 2010 b), the algorithm selects the processor with the least allocation cost. If more than one processor has the same least allocation cost, then the algorithm selects the next least allocation cost processor for the job. If again tie occurs, among the processors that has the next least allocation cost select the processor that can host minimum job workload. The algorithm reduces the processor availability time and the job workload. The process repeats until the processor availability, or the job workload becomes zero.

Suri and singh (2010) proposed a decentralized grid system model for load balancing across the grid environment. The grid system model comprising of clusters, cluster servers treated as Coordinator Nodes (CN). The cluster consists of multiple Worker Nodes (WN). WN has different processing powers. The authors proposed a dynamic load balancing algorithm for job scheduling across the decentralized grid environment. User generates the computationally intensive and mutually independents jobs. User submits the jobs directly to the CN, and the CN checks whether the cluster is overload or not. If the utilization value of all the resources in the cluster has reached a threshold value, then the cluster said to be overload. If the cluster becomes overload, then the CN contacts Grid Information Centre (GIC). GIC provides the list of underutilized clusters. CN schedules the tasks to one of the selected cluster. The load of a cluster depends on the load of each WN. The load determines based on the parameters such as CPU (Central Processing Unit) utilization, memory usage, and queue length.

Murugesan and chellappan (2009) presented a resource scheduling model. The model integrates DLT technique. The scheduling strategy divides the load equally into fractions, and the model allocates each fraction to the processor with the aim of minimizing the processing time. The disadvantage is the scheduling model does not support the dynamic nature of load and resources. The authors use the random number generation method for the division of load into equal amounts. The authors suggest that the load division depends on the resource capacity in the future.

Pop *et al*. (2008) presented a decentralized architecture, scheduling policies, and fault management to provide a solution to optimize large scale scientific application workflows by efficiently scheduling the grid resources. Scheduling policies presented to optimize the turnaround time of the dependent jobs, and the authors' have proposed approaches for failure, and the rescheduling does not consider the entire executing process.

Lee *et al*. (2010) proposed new decentralized scheduling scheme. The scheme includes local scheduling, backfilling across multiple nodes and queue balancing. The authors' demonstrated the effectiveness of the algorithm by comparing the decentralized approach with an online centralized scheduler. The decentralized approach balances the load and improves the throughput compared to that of the centralized scheduler.

In the previous work, decentralized grid architecture model as a collection of clusters was proposed. Each cluster consists of more number of WN. In the existing algorithm Decentralized Hybrid Job Scheduling Algorithm (DHJSA), the scheduling of jobs in a decentralized grid environment is done by applying DLT and Least Cost Method to select the appropriate cluster and the appropriate WN for execution. The scheduling criteria in this work do not consider the communication cost for scheduling. The algorithm well suits for computing-intensive job (Kamalam and Murali Bhaskaran, 2011a).

In the previous work, Novel Adaptive Decentralized Job Scheduling Algorithm (NADJSA), the scheduling of jobs to the WN is done by dividing the jobs into subjobs and schedules to the WN with minimum completion time. The algorithm does not consider the load of the resources and the available processing power of the resources for scheduling (Kamalam and Murali Bhaskaran, 2012).

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 171-177

This paper proposes grid architecture as a collection of clusters with multiple WN in a cluster. The proposed algorithm Best Cluster Decentralized Job Scheduling Algorithm optimizes the completion time of the job by dividing the jobs into subjobs. The subjobs are scheduled to the WN of different clusters in a decentralized grid environment. The algorithm efficiently schedules both the computing-intensive jobs and data-intensive jobs based on the best cluster value.

MATERIALS AND METHODS

Decentralized Grid Architecture Model:

A decentralized grid system model for providing efficient scheduling in grid comprising of a collection of clusters where cluster servers are treated as Coordinator Nodes (CN) represented as CN={CN_1, CN_2, … , CN_m}. Each cluster consists of multiple Worker Nodes (WN) represented as WN={WN_{i1},WN_{i2}, … ,WN_{in}}. Every WN has different processing speed. The WN within the cluster are interconnected by a local area network. The clusters are connected through a wide area network. Grid Information Centre (GIC) maintains the processing speed and memory utilization value of all the Worker Nodes in the grid. CN of each cluster provides this information to GIC periodically (Kamalam and Murali Bhaskaran, 2011b).

The user submits the job to the CN of the cluster. The set of jobs is denoted as J={J_1,J_2, … ,J_k}. Each job is divided into subjobs represented as J_i={SJ_{i1},SJ_{i2}, … , SJ_{il}}.

In a distributed grid environment, the job scheduling is a linear programming transportation problem. An efficient scheduling algorithm is essential for scheduling of jobs originating from any cluster to any other clusters in a decentralized grid environment at minimum transportation cost.

Existing Research:

A divisible job J_i is divided into subjobs to the maximum of five partitions. Let k be the number of jobs and q be the number of partitions.

Job J is denoted as J={J_1,J_2, … ,J_k}

Subjob Ji is denoted as J_i={SJ_{i1},SJ_{i2}, … , SJ_{iq}}

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 (Kamalam and Muralibhaskaran, 2011a).

Proposed Work:

The user submits the job to the CN of the cluster. While submitting the job the user specifies the type of jobs, computing-intensive jobs and data-intensive jobs. The CN selects the cluster with best cluster value and in the best cluster; the WN with highest processing power is selected and assigns the job to that WN. The computing-intensive job is divided into subjobs and scheduled to the appropriate WN in the best cluster. The data-intensive job is scheduled to a single WN whose processing power is high.

- The best value of each cluster is defined as,

$$BC_i = \alpha\ ATP_i + \beta\ APP_i$$

where,

BC_i – Best Cluster value for cluster i

α , β – weight value whose sum is equal to 1.

- For data-intensive jobs, the value of α and β are 0.7 and 0.3 respectively. For computing-intensive jobs, the value of α and β are 0.3 and 0.7 respectively.

ATP_i – Average Transmission Power of cluster i.

APP_i – Average Processing Power of cluster i.

- Average Transmission Power of cluster i is defined as:

$ATP_i = \sum_{j=1}^{n} B_{ij} / m-1,\ i \neq j$

where,

B_{ij} is the available bandwidth between the cluster i and cluster j.

m is the number of clusters in the grid system.

- Available processing power of WN_k is defined as:

$WNPower_k = WNPower_k(1-load_k)$

$WNPower_k$ is the processing power of WN_k

$load_k$ is the load of WN_k

- Current load of the WN_k is defined as :

$load_k = \sum_{i=1}^{n} Joblength_i / WNPower_k$

where,

Joblength is the length of the job submitted to the WN_k.

n is the number of jobs submitted to the $WN_{k.}$

- The Current Processing Power of WN_k is defined as:

CPP_k = ($WNPower_k$/Max (WNPower of all WN in all cluster)) * 10

CPP of the WN which has Maximum Available Processing Power is assigned as 10.

- Average Processing power of $Cluster_i$ is defined as:

$APP_i = \sum_{k=1}^{n} CPP_k / n$

where,

$CPP_{k\,-}$ current processing power of WN_k in cluster i.

n - number of WN in cluster i.

Best Cluster Decentralized Job Scheduling Algorithm (BCDJSA) is as follows:

Step 1: The initial status of the WN in each cluster is noted. The Best cluster value is calculated for all clusters. makespan is initialized to zero.
Step 2: If J is empty then goto step 11.
Step 3: If a job J_i completes the execution of all its subjobs the results are dispatched to the originated cluster and remove the job J_i from the job set J.
Step 4: If the new job arrived is a data-intensive job, the CN selects the best cluster that the highest best cluster value. In that best cluster the CN schedules the job to the WN that has the highest CPP.
Processing time = joblength + Data transfer time
Data transfer time = Data size in MB/ Bandwidth.
Step 5: If a new job arrives at CN of any cluster C_i and if the job is computing-intensive job, then divide the job into maximum of 5 equal divisions and then add subjob Sj_{i1}, ... , Sj_{in} to subjob set SJ.
Step 6: Select the cluster that has the highest best cluster value. Select the WN that has the highest CPP in that cluster and schedule the subjob to that WN.
Step 7: makespan is calculated.
Processing time=Subjob length/ Processing power of selected WN
If Processing Time is > makespan
makespan=Processing time
Step 8: Average Processing Power, Average Transmission Power, cluster value of each cluster is updated.
Step 9: Repeat Step 6-8 until all subjob is scheduled.
Step 10: Repeat Step 5-9 until the job set J is empty.
Step 11: END.

RESULTS AND DISCUSSION

The analysis of the proposed BCDJSA is compares with the existing DHJSA 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 Node per Cluster	10
Processing Power of Worker Node	500-5000 MIPS
Computing-intensive job	$\alpha = 0.2, \beta = 0.8$
Data-intensive job	$\alpha = 0.8, \beta = 0.2$
Job Length	2,50,000–6,50,000 MI
Number of jobs	100-1000

Simulation results:

The performance measure of the job scheduling algorithm is based on the three parameters: Total processing time called the Makespan, Number of computing-intensive job, and the Number of data-intensive job.

The performance of the proposed algorithm BCDJSA is measured for three different cases.

Case 1: Equal number of data-intesive job and computing-intensive job.

Case 2: More number of data-intesive job and less number of computing-intensive job.

Case 3: Less number of data-intesive job and more number of computing-intensive job.

Table 2 show makespan values obtained by DHJSA for three different cases.

Table 3 show makespan values obtained by the proposed algorithm BCDJSA for three different cases.

Figure 1, 2 and 3 shows the graphical representation the makespan values of all the three different cases for the existing algorithm DHJSA and the proposed algorithm BCDJSA.

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 171-177

Table 2: Makespan obtained by DHJSA.

No. of Jobs	Case1	Case2	Case2
100	23846	25589	20234
200	84746	59149	41502
300	168033	149819	132186
400	214860	195943	183941
500	289638	268924	249186

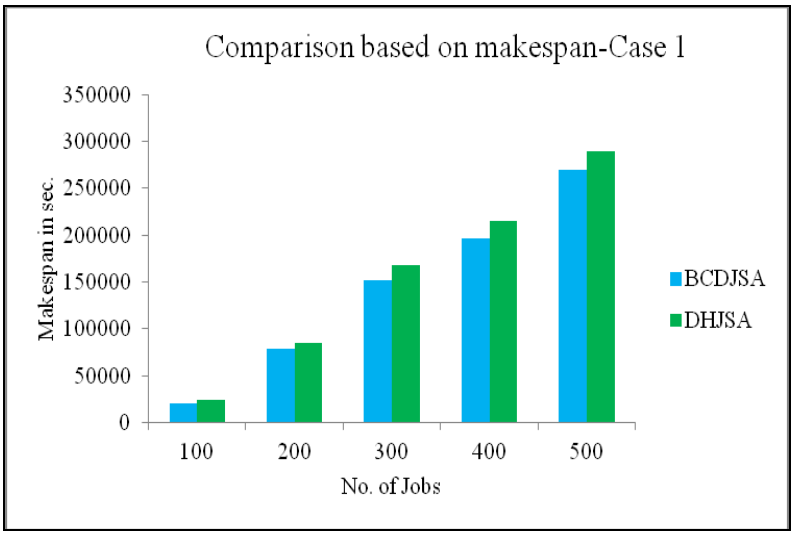

Fig. 1: Comparison based on makespan for Case 1.

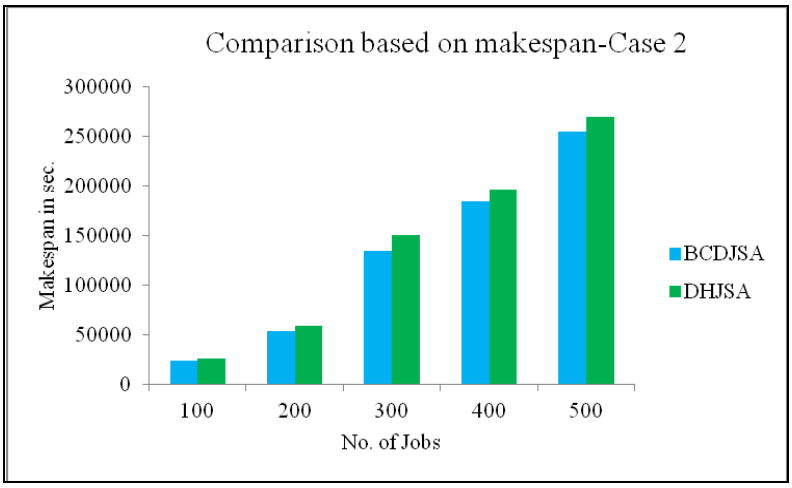

Fig. 2: Comparison based on makespan for Case 2.

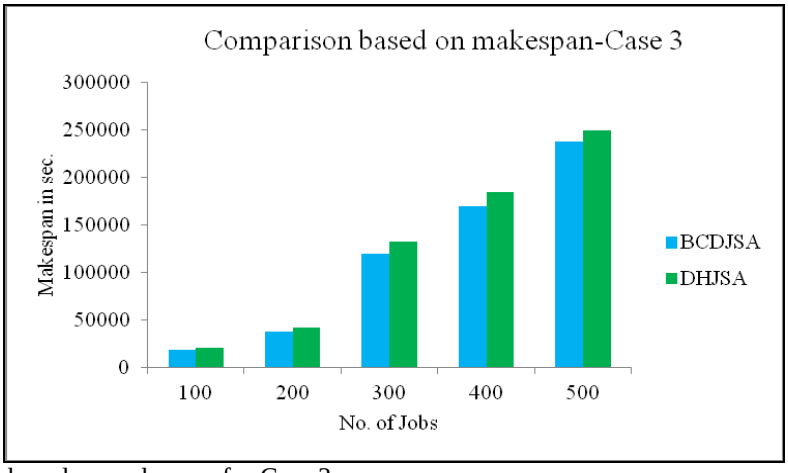

Fig. 3: Comparison based on makespan for Case 3.

Table 3: Makespan obtained by BCDJSA.

No. of Jobs	Case1	Case2	Case2
100	20953	23431	18923
200	78300	53082	37411
300	151406	134381	119423
400	196349	184296	169872
500	269142	254309	237214

The simulation result shows that the proposed BCDJSA provides a better makespan than the DHJSA.

Conclusion and Future Work:

In this paper, an efficient job scheduling algorithm for a decentralized grid environment is proposed. The proposed BCDJSA allocates both computing-intensive jobs and data-intensive jobs efficiently.

BCDJSA selects the best worker node to execute a job according to the status of the worker nodes in every cluster. The update rules are applied to record the newest status of each WN. The CN uses the newest information to assign the next job.

The proposed algorithm aims at scheduling with minimum time (processing time and transfer time). The BCDJSA achieves makespan than the DHJSA. The results show that the proposed BCDJSA reduces the makespan, improves the resource utilization and balances the load across the grid environment.

In future, the proposed BCDJSA can be improved to handle replica strategy in data-intensive jobs. In this paper, the algorithm schedules the independent jobs, but the jobs may have precedence relations. The proposed BCDJSA algorithm can be extended to schedule dependent jobs in the future.

REFERENCES

Baker, M., R. Buyya and D. Lafornza, 2012. Grids and Grid Technologies for Wide area Distributed Computing. Software Practice and Experience, 32(15): 1437-1466.

Caron, E., V. Garonne and A.T. Saregorodtsev, 2007. Definition, modelling and Simulation of a grid computing scheduling system for high throughput computing. Future Generation Computer Systems, 968-976.

Chang, R.S., J.S. Chang and P.S. Lin, 2009. An Ant algorithm forbalanced job scheduling in Grids. Future Generation Computer Systems, 20-27.

Foster, I. and C. Kesselman, 1999. The GRID: Blueprint for a New Computing Infrastructure. Morgan, Kaufmann.

Garey, M.R. and D.S. Johnson, 1979. Computers and Intractability:A Guide to the theory of NP-Completeness. Freeman,CA.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2011a. An effective approach to job scheduling in decentralized grid environment. International Journal of Computer Applicatons, DOI:10.5120/2914-3838: 26-30.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2011b. An Efficient Hybrid Job Scheduling for Computational Grids. International Journal of Computer Applications.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2012. Novel Adaptive Job Scheduling algorithm on Heterogeneous Grid Resources. American Journal of applied Sciences, 1294-1299.

Lee, J., P. Keleher and A. Sussman, 2010. Decentralized Dynamic Scheduling across Heterogeneous Multi-core Desktop Grids. IEEE.

Murugesan, G. and C. Chellappan, 2009. An Economical Model for Optimal Distribution of Loads for Grid Applications. International Journal of Computer and Network Security, 1(1).

Pop, F., C. Dobre and V. Cristea, 2008. Decentralized Dynamic Resource Allocation for Workflows in Grid Environments. IEEE 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, DOI 10.1109/SYNASC.2008.15:557-563.

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

Shah, S.N.M., A.K.B. Mahmood and A. Oxley, 2010a. Hybrid Resource Allocation for Grid Computing. IEEE 2nd International Conference on Computer Research and Development, 426-431.

Shah, S.N.M., A.K.B. Mahmood and A. Oxley, 2010b. Modified Least Cost Method for Grid Resource Allocation. IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, DOI:10.1109/CyberC., 47: 218-225.

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, DOI: 10.1109/ADCC 2010.5423048: 10-13.

Tierney, B., W. Johnson, J. Lee and M. Thompson, 2000. A data intensive distributed computing architecture for grid applications. Future Generation Computer Systems, 473-48.

Venugopal, S., R. Buyya and K. Ramamohanrao, 2006. A Taxonomy of data grids for distributed data sharing, management and processing. ACM computing surveys, 1-8.

Wei, G., Y. Ling, A.V. Vasilakos, B. Xiao and Y. Zheng, 2010. PIVOT: An adaptive information discovery framework for computational grids. Information sciences: 4543-4556.

Xiao, Q., 2007. Design and analysis of a load balancing strategy in grids. Future Generation Computer Systems, 132-137.