A^{2}DLT: Divisible Load Balancing Model for Scheduling Communication-Intensive Grid Applications

M. Othman*, M. Abdullah, H. Ibrahim, and S. Subramaniam

Department of Communication Technology and Network,
University Putra Malaysia, 43400 UPM Serdang, Selangor D.E., Malaysia
mothman@fsktm.upm.edu.my, monabdullah@hotmail.com

Abstract. Scheduling an application in data grid is significantly complex and very challenging because of its heterogeneous in nature of the grid system. Divisible Load Theory (DLT) is a powerful model for modelling data-intensive grid problem where both communication and computation loads are partitionable. This paper presents a new divisible load balancing model known as adaptive ADLT (A^{2}DLT) for scheduling the communication intensive grid applications. This model reduces the maximum completion time (makespan) as compared to the ADLT and Constraint DLT (CDLT) models. Experimental results showed that the model can balance the load efficiently, especially when the communication-intensive applications are considered.

Keywords: Divisible Load Theory, Data Grid, Load Balancing.

1 Introduction

In data grid environment, many large-scale scientific experiments and simulations generate very large amounts of data in the distributed storages, spanning thousands of files and data sets [1]. Due to the heterogenous nature of the grid system, scheduling an applications in such environment either data- or communication- intensive is significantly complex and challenging. Grid scheduling is defined as a process of making scheduling decision involving allocating job to resources over multiple administrative domains [2].

The DLT has emerged as a powerful model for modelling data-intensive grid problem [3]. The DLT model exploits the parallelism of a divisible application which is continuously divisible into parts of arbitrary size, by scheduling the loads in a single source onto multiple computing resources. The load scheduling in data Grid is addressed using DLT model with additional constraint that each worker node receives the same load fraction from each data source [4]. Most of the previous models do not take into account the communication time. In

* The author is also an associate researcher at the Lab of Computational Science and Informatics, Institute of Mathematical Research (INSPEM), University Putra Malaysia.

M. Bubak et al. (Eds.): ICCS 2008, Part I, LNCS 5101, pp. 246–253, 2008.

order to achieve a high performance, we must consider both communication- and computation-times [5,6].

In [7], the CDLT is used for scheduling decomposable data-intensive applications and the results are compared with the results of genetic algorithm. The same constraint is tested, which was suggested in [4] and each worker node receives the same load fraction from each data source. They considered the communication time but not in dividing the load. Firstly, they divided the load using DLT model then added the communication time to the makespan. When ADLT model is proposed for scheduling such applications and compared with CDLT model, and it gives better performance [8].

In this paper, a A^{2}DLT model is proposed as an improvement of ADLT model. The objective of the model is to distribute loads over all sites in such a way to achieve an optimal makespan for large scale jobs.

2 Scheduling Model

In [5], the target data intensive application model can be decomposed into multiple independent subtasks and executed in parallel across multiple sites without any interaction among subtasks. Lets consider job decomposition by decomposing input data objects into multiple smaller data objects of arbitrary size and processing them on multiple virtual sites. High Energy Physic (HEP) jobs are arbitrarily divisible at event granularity and intermediate data product processing granularity [1]. In this research, assuming that a job requires a very large logical input data set (D) consists of N physical datasets and each physical dataset (of size L_k) resides at a data source (DS_k, for all $k = 1, 2, \ldots, N$) of a particular site. Fig 1 shows how the logical input data (D) is decomposed onto networks and their computing resources.

The scheduling problem is to decompose D into datasets (D_i for all $i = 1, 2, \ldots, M$) across M virtual sites in a Virtual Organization (VO) given its initial physical decomposition. Again, we assume that the decomposed data can be analyzed on any site.

2.1 Notations and Definitions

All notations and their definitions used throughout this paper are shown in Table 1.

2.2 Cost Model

The execution time cost (T_i) of a subtask allocated to the site i and the turn around time ($T_{Turn\_Around\_Time}$) of a job can be expressed as follows

$$T_i = T_{input\_cm}(i) + T_{cp}(i) + T_{output\_cm}(i, d)$$

and

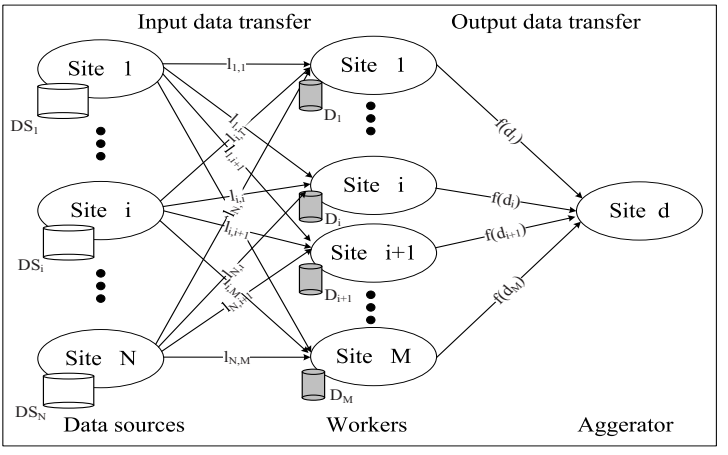

Fig. 1. Data decomposition and their processing

Table 1. Notation and Definition

Notation	Definition
M	The total number of nodes in the system
N	The total number of data files in the system
L_i	The loads in data file i
L_{ij}	The loads that node j will receive from data file i
L	The sum of loads in the system, where $L = \sum_{i=1}^{N} L_i$
α_{ij}	The amount of load that node j will receive from data file i
α_j	The fraction of L that node j will receive from all data files
w_j	The inverse of the computing speed of node j
Z_{ij}	The link between node i and data source j
T_i	The processing time in node i

$$T_{Turn\_Around\_Time} = \max_{i=1}^{M}\{T_i\},$$

respectively. The input data transfer ($T_{input\_cm}(i)$), computation ($T_{cp}(i)$), and output data transfer to the client at the destination site d ($T_{output\_cm}(i,d)$) are presented as a $\max_{k=1}^{N}\{l_{ki} \cdot \frac{1}{Z_{ki}}\}$, $d_i \cdot w_i \cdot ccRatio$ and $f(d_i) \cdot Z_{id}$, respectively. The Z_{ij} is the network bandwidth between site i and j, w_i is the computing time to process a unit dataset of size 1MB at site i, the function $f(d_i)$ is an output data size and *ccRatio* is the non-zero ratio of computation and communication. The turn around time of an application is the maximum among all the execution times of the sub tasks.

The problem of scheduling a divisible job onto M sites can be stated as deciding the portion of original workload (D) to be allocated to each site, that is, finding a distribution of l_{ki} which minimizes the turn around time of a job.

The proposed model uses this cost model when evaluating solutions at each generation.

3 ADLT Scheduling Model

In all literature related to the divisible load scheduling, an optimality criterion [6] was used to derive an optimal solution. In order to obtain an optimal makespan, it is necessary and sufficient that all the sites that participate in the computation must complete at the same time. Otherwise, load could be redistributed to each sites and this will improve the processing time. This optimality principle in the design of load distribution strategy is used. The communication time fraction is added into the ADLT model and the final fraction of the model is shown as below,

$$CM_{i,j} = \frac{1}{w_j(\Sigma_{x=1}^{M}\frac{1}{w_x})} + \frac{1}{Z_{i,j}\sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{Z_{x,y}}} \tag{1}$$

$$\alpha_j = \frac{CM_{i,j}}{\sum_{i=1}^{N}\sum_{j=1}^{M} CM_{i,j}} \tag{2}$$

and

$$\alpha_{i,j} = \frac{CM_{i,j}}{\sum_{i=1}^{N}\sum_{j=1}^{M} CM_{i,j}} L_i. \tag{3}$$

Details of this model and their derivation can be found in [8].

4 Proposed A^{2}DLT Model

In the ADLT model, the fraction equations (1), (2) and (3) are taken separately from each source, see [8]. In addition the node speed- and link speed- fractions are also taken separately, thus yields the node speed fraction as,

$$\frac{\frac{1}{w_j}}{(\Sigma_{x=1}^{M}\frac{1}{w_x})} \tag{4}$$

while the link speed fraction for each link given as,

$$\frac{\frac{1}{Z_{i,j}}}{\sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{Z_{x,y}}}. \tag{5}$$

Again, the summation of these fractions are also taken separately at each source. These loads are divided by using these fractions and finally the makespan is calculated. In the proposed model, we must balance the load of the whole system (means all sources). In other word, the node speed fraction is calculated separately, thus yield the node speed- and the link speed- fractions given as,

$$\frac{\frac{1}{w_j}}{(\Sigma_{x=1}^{M}\frac{1}{w_x})+\sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{Z_{x,y}}} \tag{6}$$

and,

$$\frac{\frac{1}{Z_{i,j}}}{(\Sigma_{x=1}^{M}\frac{1}{w_x})+\sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{Z_{x,y}}}, \tag{7}$$

respectively. Finally, the new fraction is given as,

$$CM_{i,j}=\frac{\frac{1}{w_j}}{(\Sigma_{x=1}^{M}\frac{1}{w_x})+\sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{Z_{x,y}}}+\frac{\frac{1}{Z_{i,j}}}{(\Sigma_{x=1}^{M}\frac{1}{w_x})+\sum_{x=1}^{N}\sum_{y=1}^{M}\frac{1}{Z_{x,y}}}, \tag{8}$$

$$\alpha_j=\frac{CM_{i,j}}{\sum_{i=1}^{N}\sum_{j=1}^{M}CM_{i,j}}, \tag{9}$$

and

$$\alpha_{i,j}=\frac{CM_{i,j}}{\sum_{i=1}^{N}\sum_{j=1}^{M}CM_{i,j}}L_i. \tag{10}$$

5 Numerical Experiments

To measure the performance of the proposed A^{2}DLT model against the previous models, randomly generated experimental configurations are used, see [7,8]. The network bandwidth between sites is uniformly distributed between 1Mbps and 10Mbps. The location of n data sources (DS_k) is randomly selected and each physical dataset size (L_k) is randomly selected with a uniform distribution in the range of 1GB to 1TB. We assumed that the computing time spent in a site i to process a unit dataset of size 1MB is uniformly distributed in the range $1/r_{cb}$ to $10/r_{cb}$ seconds where r_{cb} is the ratio of computation speed to communication speed.

We examined the overall performance of each model by running them under 100 randomly generated Grid configurations. We varied the parameters, *ccRatio* (0.001 to 1000), M (20 to 100), N (20 to 100), r_{cb} (10 to 500) and data file size (1 GB to 1 TB). When both the number of nodes and the number of data files are 50, the results are collected and shown in Fig. 2.

The results showed that the makespan of the proposed model is better than the other models, especially when the *ccRatio* is less than 1 (communication-intensive applications). Thus, the proposed model balances the load among the nodes more efficiently.

From Table 2, the results show that the A^{2}DLT is 34% better than CDLT in terms of makespan. While the A^{2}DLT is better than ADLT by 25%. These results showed that A^{2}DLT is the best among CDLT and ADLT models.

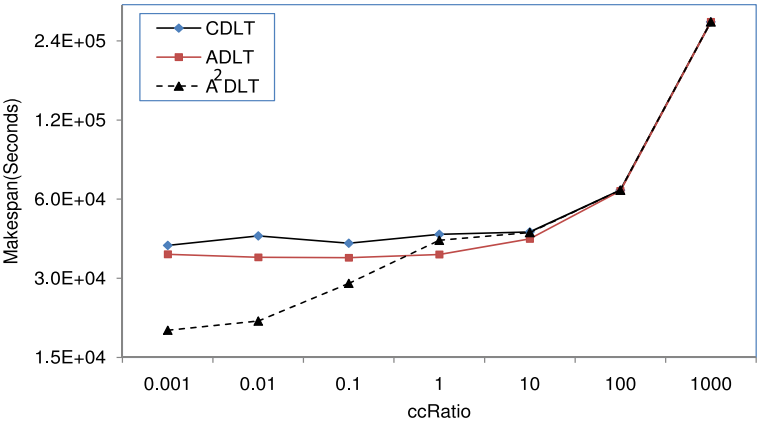

Fig. 2. Makespan for A^{2}DLT, ADLT and CDLT models (N=50, M=50 and *ccRatio*=0.001 to 1000)

Table 2. Percentage makespan improvements of A^2*DLT* against CDLT and ADLT models

ccRatio	CDLT (%)	ADLT (%)
0.001	49	49
0.01	53	43
0.1	30	20
1	5	-13
Average	34	25

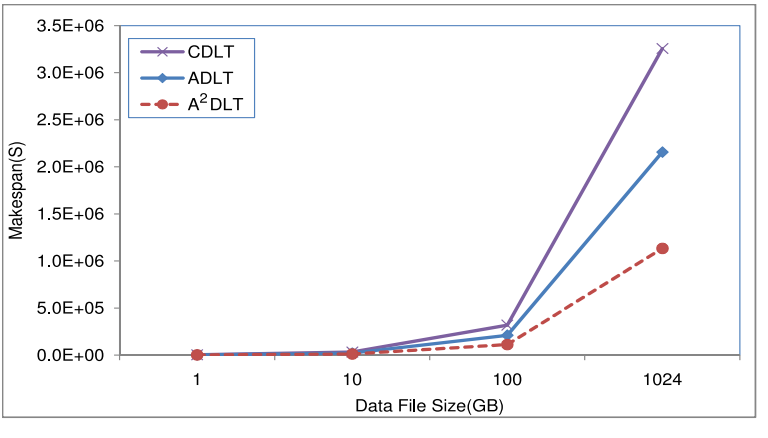

Fig. 3. Makespan vs. Data file Size for A^{2}DLT, ADLT and CDLT models (N=100, M=100 and *ccRatio*=0.001)

When we compare the A^{2}DLT model to the CDLT and ADLT with different size of data file, the A^{2}DLT model produces a better result as increasing the size of data file. The result is shown in Fig. 3.

The impact of the ratio of output data size to input data size is also shown in Fig. 4. The A^{2}DLT model performs better for communication intensive applications that generate small output data compared to input data size (low *oiRatio*). For computation intensive applications, the ratio of output data size to input data size does not affect the performance of the algorithms much unless when *ccRatio* is 1000.

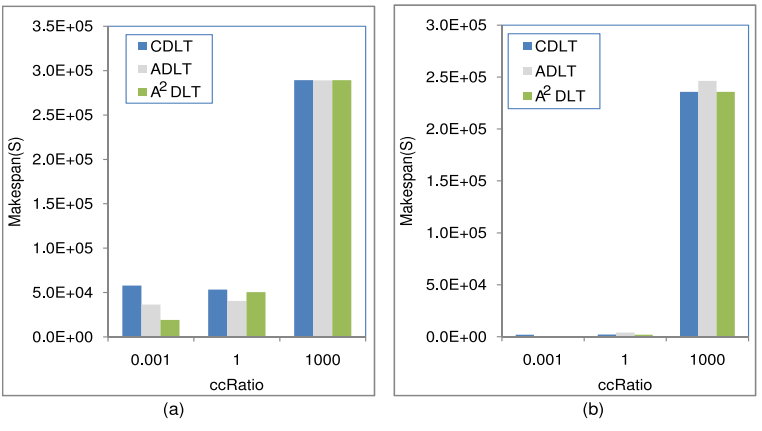

Fig. 4. The impact of output data size to input data size (a)$oiRatio > 0.5$(b)$oiRatio = 0$: No output or small size of output

6 Conclusion

Previously, ADLT model reduced the makespan for scheduling divisible load application. In this paper, an improvement version of ADLT model known as the A^{2}DLT model is proposed. The new model reduces the makespan and balance the load better than ADLT model, especially for communication-intensive applications. The experiment results showed that A^{2}DLT model improved with an average of 34% and 25% of makespan compared to CDLT and ADLT models, respectively. With such improvement, the proposed model can be integrated in the existing data grid schedulers in order to improve the performance.

References

1. Jaechun, N., Hyoungwoo, P.: GEDAS: A Data Management System for Data Grid Environments. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2005. LNCS, vol. 3514, pp. 485–492. Springer, Heidelberg (2005)

2. Venugopal, S., Buyya, R., Ramamohanarao, K.: A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing. ACM Computing Surveys 38(1), 1–53 (2006)
3. Robertazzi, T.G.: Ten Reasons to Use Divisible Load Theory. IEEE Computer 36(5), 63–68 (2003)
4. Wong, H.M., Veeravalli, B., Dantong, Y., Robertazzi, T.G.: Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints. In: Proceeding of the IASTED Conference on Parallel and Distributed Computing and Systems, Marina del Rey, USA (2003)
5. Mequanint, M.: Modeling and Performance Analysis of Arbitrarily Divisible Loads for Sensor and Grid Networks. PhD Thesis. Dept. Electrical and Computer Engineering, Stony Brook University, New York USA (2005)
6. Bharadwaj, V., Ghose, D., Robertazzi, T.G.: Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. Cluster Computing 6, 7–17 (2003)
7. Kim, S., Weissman, J.B.: A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications. In: Proceeding of the International Conference on Parallel Processing. IEEE Computer Society Press, Washington (2004)
8. Othman, M., Abdullah, M., Ibrahim, H., Subramaniam, S.: Adaptive Divisible Load Model for Scheduling Data-Intensive Grid Applications. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4487, pp. 446–453. Springer, Heidelberg (2007)