American Journal of Applied Sciences 6 (4): 626-630, 2009
ISSN 1546-9239


# Closed form Solution for Scheduling Arbitrarily Divisible Load Model in Data Grid Applications: Multiple Sources

Monir Abdullah, Mohamed Othman, Hamidah Ibrahim and Shamala Subramaniam
Department of Communication Technology and Network,
University Putra Malaysia, 43400 UPM Serdang, Selangor DE, Malaysia

**Abstract:** Scheduling an application in data grid was significantly complex and very challenging because of its heterogeneous in nature of the grid system. When the Divisible Load Theory (DLT) model had emerged as a powerful model for modeling data-intensive grid problem, Task Data Present (TDP) model was proposed based on it. This study presented a new Adaptive TDP (ATDP) for scheduling the intensive grid applications. New closed form solution for obtaining the load allocation was derived while computation speeds and communication links are heterogeneous. Experimental results showed that the proposed model can balance the load efficiently.

**Key words:** Data grid, divisible load theory

## INTRODUCTION

Grid computing applies the resources of many computers in a network to a single problem at a given time-usually to a scientific or computational problem that requires a greater number of CPU cycles or access to large amounts of data[1]. 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[2]. Due to the heterogeneous in nature of the grid system, scheduling an applications in such environment either data- or communication intensives is significantly complex and challenging. Grid scheduling is defined as the process of making scheduling decision involving allocating job to resources over multiple administrative domains[3].

A decoupled scheduling architecture for data intensive applications is also proposed[7]. In this research, they proposed Task Data Present (TDP) model. The results show that when the job is scheduled to a site where the data is available the data transfer is minimal but the response time suffers when there is no data replication. This is because a few sites which host the data are overloaded in this case and hence, making a case for dynamic replication of data.

Recently, DLT model has emerged as a powerful model for modeling data-intensive grid problem[4]. DLT 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[5].

However, most of the previous models do not take into account the communication time. Whereas, in order to achieve a high performance, we must consider both communication and computation time[6].

In addition, a load balancing algorithm is also developed on a structure of data of network type WAN, which guarantees its portability on any grid computing. The distribution of loads assures the convergence of the algorithm in an acceptable time[9]. In this research, the communication time is not considered.

TDP model was examined with other strategies for non-divisible applications[7,10]. It was modified to be able to schedule divisible load applications[8]. This strategy maps tasks only to the sites where the required data is present. Each task processes the data sets residing at that site. There is no input data transfer in this case. They considered the communication time but not in dividing the load. Instead, they divided the load using DLT model then added the communication time to the makespan.

Our previous research in this area is the Adaptive DLT (ADLT) for scheduling the arbitrarily divisible load in data grid application[12]. In this research, the communication time and communication time are considered jointly.

**Corresponding Author:** Monir Abdullah, Mohamed Othman, Department of Communication Technology and Network, University Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
Tel: +603-89466535 Fax: +603-89466577

In this study, a new Adaptive TDP model is proposed as an improvement of TDP model. The objective is to find the optimal allocation of workload of M processing nodes by the examination of the effect of non-negligible communication delay as well as computation time and the network configurations. The design of our proposed model adopts the divisible load paradigm, referred to as the Divisible Load Theory (DLT), which is shown to be efficient in handling large volume loads.

**Scheduling model:** A generic data grid computing system infrastructure considered here comprises a network of supercomputers and/or clusters of computers connected by Wide Area Network (WAN), having different computational and communication capabilities. We consider the problem of scheduling large-volume loads (divisible loads) within in multiple sites. Communication is assumed to be predominant between such cluster nodes and is assumed to be negligible within a cluster node[5,8,11].

The target data intensive application model can be decomposed into multiple independent subtasks and executed in parallel across multiple sites without any interaction among sub tasks. For example, let's 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[2]. 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…, N) of a particular site. Figure 1 shows how the logical input data (D) is decomposed onto networks and their computing resources.

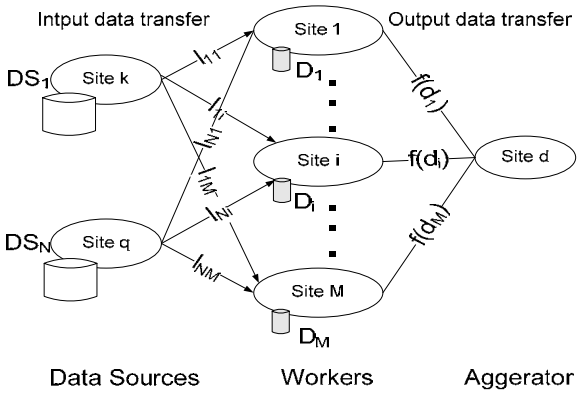


Fig. 1: Data decomposition and their processing

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

For the notations, definitions that used in this research are stated in Table 1.

The execution time of a subtask allocated to the site i ($T_i$) and the turn around time of a job J (Tturn_around_time) can be expressed as follows:

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

$$T_{turnaround\_time} = \max_{i=1,...,M}\{T_i\}$$

The cost ($T_i$) includes input data transfer ($T_{input\_cm}(i)$), computation ($T_{cp}(i)$) and output data transfer to the client at the destination site d ($T_{output\_cm}(i,d)$):

$$T_{input\_cm}(i) = \max_{k=1..m}\{\alpha_{ki} \cdot \frac{1}{z_{ki}}\}$$

$$T_{cp}(i) = d_i \cdot w_i$$

$$T_{output\_cm}(i,d) = f(d_i) \cdot z_{id}$$

We assume that data from multiple data sources can be transferred to a site i concurrently in the wide area environment and computation starts only after the assigned data set is totally transferred to the site. Hence, the problem of scheduling a divisible job onto n sites can be stated as deciding the portion of original workload (D) to be allocated to each site, that is, finding a distribution of distribution of $\{\alpha_{ki}\}$ which minimizes the turn-around time of a job. The proposed SA approach uses this cost model when evaluating solutions at each generation.

**TDP scheduling model:** Firstly, TDP model was proposed for scheduling indivisible load[7]. Consequently, it is modified to be work on scheduling divisible load applications[7]. This strategy maps tasks only to the sites where the required data is present. Each task processes the data sets residing at that site.

Table 1: Terminology, definitions and notations

| | |
|---|---|
| N | The total number of data files in the system |
| M | The total number of nodes in the system |
| $L_i$ | The loads in data file i |
| $L_{ij}$ | The loads that node i will receive from data file j |
| L | The sum of loads in the system, where L = $\sum_{i=1}^{N} L_i$ |
| $\alpha_{ij}$ | The fraction of L that node i will receive from all data file j |
| $w_i$ | The inverse of the computing speed of node i |
| $Z_{ij}$ | The link between node i and data source j |
| T(i) | The processing time in node i |

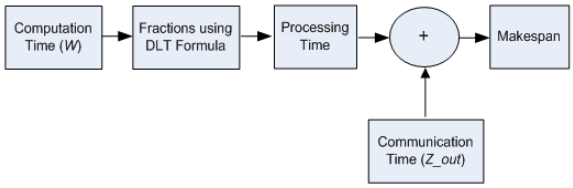


Fig. 2: Framework of TDP model

There is no input data transfer in this case. In the beginning, they calculate the processing time using DLT model that was proposed in[4]. Then, the transfer output time is added to calculate the makespan. The equation of calculating the load fraction is:

$$\frac{(\frac{1}{w_j})}{\sum_{x=1}^{M}(\frac{1}{w_x})} \quad (1)$$

So the amount of load that site i gives site j is calculated as:

$$\alpha_{i,j} = \frac{(\frac{1}{w_j})}{\sum_{x=1}^{M}(\frac{1}{w_x})} L_i \quad (2)$$

To understand how the TDP model works, the framework is shown in Fig. 2.

**Proposed ATDP scheduling model:** In the previous TDP model, the loads were divided by using the DLT model and finally the makespan was calculated. In the proposed model, we try to balance the loads by considering communication time. In other word, the node speed fraction was calculated together with the communication time fraction as follows.

The communication time fraction is added into the ATDP model. The fraction of load if we consider the computation time only is:

$$\frac{(\frac{1}{w_j})}{\sum_{x=1}^{M}(\frac{1}{w_x})} \quad (3)$$

and if we consider the communication time only (time for transferring the processed load to the output node), the load fraction will be as:

$$\frac{(\frac{1}{z_j})}{\sum_{x=1}^{M}(\frac{1}{z_x})} \quad (4)$$

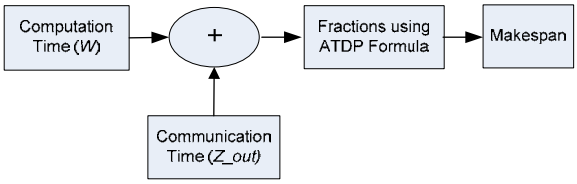


Fig. 3: Framework of ATDP model

In the proposed model, we will consider both-the communication time fraction as well as the computation time fraction. Thus, the Combination Fraction (CF) will be as:

$$CF_{ij} = \frac{\frac{1}{w_j}}{\sum_{x=1}^{M}\frac{1}{w_x}} + \frac{\frac{1}{z_j}}{\sum_{x=1}^{M}\frac{1}{z_x}} \quad (5)$$

$$\alpha_j = \frac{CF_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M}CF_{ij}} \quad (6)$$

And the closed form solution is:

$$\alpha_j = \frac{CM_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M}CM_{ij}} L_i \quad (7)$$

To be clear the framework of the ATDP model is shown in Fig. 3.

## RESULTS AND DISCUSSION

To measure the performance of the proposed SA-based approach against CDLT and GA approaches, randomly generated experimental configurations were used. We made the simulation program using C++ language. The estimated expected execution time for processing a unit dataset on each site, the network bandwidth between sites, input data size and the ratio of output data size to input data size were randomly generated with uniform probability over some predefined ranges. The network bandwidth between sites is uniformly distributed between 1 Mbps and 10 Mbps.

The location of m 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. It is 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.

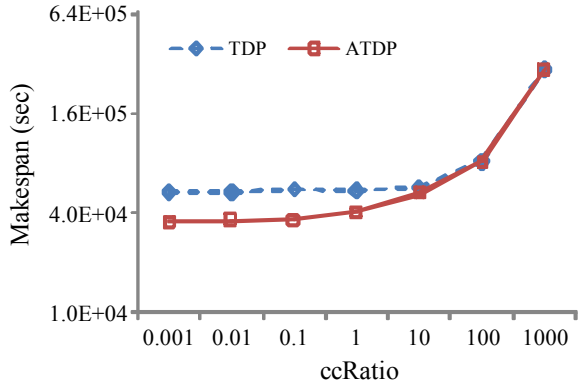


Fig. 4: Makespan for the TDP and ATDP models

Table 2: Percentage makespan improvements of ATDP model against TDP model with different ccRatio

| ccRatio | TDP (%) |
|---|---|
| 0.001 | 34 |
| 0.01 | 33 |
| 0.1 | 35 |
| 1 | 26 |
| 10 | 7 |
| 100 | 0 |
| 1000 | 0 |
| Average | 20 |

We examined the overall performance of each model by running them under 100 randomly generated Grid configurations. These parameters are varied: ccRatio (0.001-1000), M (20-100), N (20-100), $r_{cb}$ (10-500) and data file size (1 GB-1 TB). When the number of nodes and the number of data files are both 100, the results are collected and shown in Fig. 4.

The results showed that the makespan of the proposed model is better than the previous model, especially when the ccRatio is less than 1 (communication-intensive applications).

In summary, the model balances the load among the nodes more efficiently. The percentage makespan improvements of ATDP model against TDP model are clearly demonstrated in Table 2.

From Table 2, it was found that ATDP model is 20% better than TDP model in terms of makespan. These results showed that ATDP is the best model which means by applying this model will balance the load efficiently for communication intensive application.

When we compare the ATDP model to the TDP model with different size of data files, the ATDP model produces better result as compared to TDP model. The performance of ATDP model is improved when the size of data file is increased. The result is shown in Fig. 5.

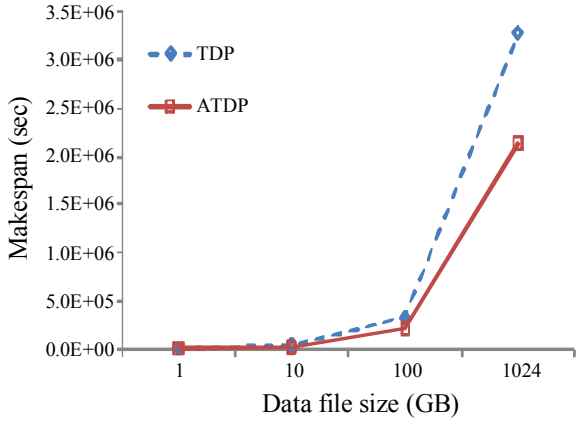


Fig. 5: Makespan vs. data file size for the TDP and TDP models

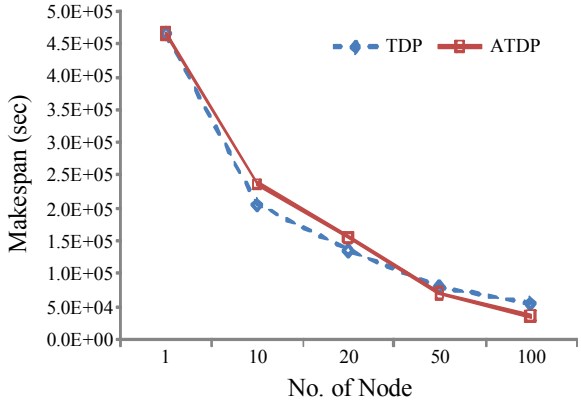


Fig. 6: Makespan vs. no. of node ATDP and TDP models (N = 100, M = 100 and ccRatio = 0.001)

Table 3: Percentage makespan improvements of ATDP model against TDP model with different data file size

| ccRatio | TDP (%) |
|---|---|
| 1 | 0 |
| 10 | -15 |
| 20 | -13 |
| 50 | 12 |
| 100 | 35 |
| Average | 4 |

Furthermore, the graphs are also plotted for makespan against number of processing node as in Fig. 6.

Figure 6, shows that ADLT model is better than TDP model when the number of processing nodes is less than 50. When the number of the processing nodes becomes 50 and above, the ATDP is produce better results. In average, the ATDP model is better than TDP model by 4%, Table 3.

Table 3 shows that, ATDP model produce in average better results than TDP model. It is also clear that the proposed the ATDP model produces better results when the number of processing node is increased.

## CONCLUSION

In this study, an improvement version of TDP model called ATDP model proposed. The ATDP model reduces the makespan and balances the load more efficiently than the TDP model. The experiment results showed that ATDP model improved with an average of 20% of the makespan compared to TDP model. With such improvement, the proposed model can be integrated in the existing data grid schedulers in order to improve the performance.

## REFERENCES


1. Richard ,R.J.A., A.J. Ajay and C. Eswaran, 2008. Implementation of computational grid services in enterprise grid environments. Am. J. Applied Sci., 5:1442-1447. http://www.articlearchives.com/computing-information-technology/computer-networks/1885216-1.html.
2. Jaechun, N. and P. Hyoungwoo, 2005. GEDAS: A data management system for data grid environments. Lecture Notes Comput. Sci., 3514:485-492. http://www.springerlink.com/content/lw2t4dwv3thw3b86/.
3. Venugopal, S., R. Buyya and K. Ramamohanarao, 2006. A taxonomy of data grids for distributed data sharing, management and processing. ACM Comput.Surv.,38:1-53. http://portal.acm.org/citation.cfm?id=1132952.1132955.
4. Robertazzi, T.G., 2003. Ten reasons to use divisible load theory. IEEE Comput., 36: 63-68. DOI: 10.1109/MC.2003.1198238.
5. Wong, H.M., B. Veeravalli, Y. Dantong and T.G. Robertazzi, 2003. Data intensive grid scheduling: Multiple sources with capacity constraints. Proceeding of the IASTED Conference on Parallel and Distributed Computing and Systems, Nov. 3-5, Marina del Rey, USA, pp:7-11. http://www.actapress.com/PaperInfo.aspx?PaperID=13753&reason=500.
6. Bharadwaj, V., D. Ghose and T.G. Robertazzi, 2003. Divisible load theory: A new paradigm for load scheduling in distributed systems. Clust. Comput.,6: 7-17. DOI: 10.1023/A:1020958815308.
7. Ranganathan, K. and I. Foster, 2002. Decoupling computation and data scheduling in distributed data-intensive applications. Proceeding of the 11th IEEE International Symposium on High Performance Distributed Computing, July 24-26, IEEE Computer Society Washington, DC., USA., pp:352. http://portal.acm.org/citation.cfm?id=823346.
8. Kim, S. and J. B. Weissman, 2004. A genetic algorithm based approach for scheduling decomposable data grid applications. Proceeding of the International Conference on Parallel Processing, Aug. 15-18, IEEE Computer Society Press, Washington DC., USA., pp: 406-413. DOI: 10.1109/ICPP.2004.9.
9. Boukerram, A. and S.A.K. Azzou, 2006. Implementation of load balancing algorithm in a grid computing. Am. J. Applied Sci., 3: 810-1813. http://www.articlearchives.com/computing-information-technology/distributed-computing/761515-1.html.
10. Takefusa, A., O. Tatebe, S. Matsuoka and Y. Morita, 2003. Performance analysis of scheduling and replication algorithms on grid datafarm architecture for high energy physics applications. Proceedings on the 12th IEEE International Symposium on High Performance Distributed Computing, June 22-24, IEEE Computer Society Press, Washington DC., USA., pp:34. http://portal.acm.org/citation.cfm?id=822087.823402.
11. Viswanathan, S., B. Veeravalli and T.G. Robertazzi, 2007. Resource-aware distributed scheduling strategies for large-scale computational cluster/grid systems. IEEE Trans. Parall. Distribut. Syst.,18:1450-1461. DOI: 10.1109/TPDS.2007.1073.
12. Othman, M., M. Abdullah, H. Ibrahim and S. Subramaniam, 2007. Adaptive divisible load model for scheduling data-intensive grid applications: Computational science. Lecture Notes Comput. Sci., 4487: 446-453. DOI: 10.1007/978-3-540-72584-8_59.