Future Generation Computer Systems 26 (2010) 971–978

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Optimal workload allocation model for scheduling divisible data grid applications

Monir Abdullah [a,*], Mohamed Othman [b], Hamidah Ibrahim [b], Shamala Subramaniam [b]

[a] Department of Computer Science, Thamar University, Thamar, Yemen

[b] Department of Communication Technology and Network, University Putra Malaysia, 43400 UPM Serdang, Selangor D.E., Malaysia

ARTICLE INFO

Article history:
Received 18 February 2009
Received in revised form
11 April 2010
Accepted 16 April 2010
Available online 5 May 2010

Keywords:
Scheduling
Divisible load theory
Data grid

ABSTRACT

In many data grid applications, data can be decomposed into multiple independent sub-datasets and distributed for parallel execution and analysis. This property has been successfully employed using Divisible Load Theory (DLT), which has been proved a powerful tool for modeling divisible load problems in data-intensive grids. There are some scheduling models that have been studied but no optimal solution has been reached due to the heterogeneity of the grids. This paper proposes a new model called the Iterative DLT (IDLT) for scheduling divisible data grid applications. Recursive numerical closed form solutions are derived to find the optimal workload assigned to the processing nodes. Experimental results show that the proposed IDLT model leads to a better solution than other models (almost optimal) in terms of makespan.

1. Introduction

Grid computing has become a promising technology for providing seamless access to heterogeneous resources and achieving a high performance benefit in wide-area environments [1]. In the last decade, data grids have increasingly become popular for a wide range of scientific and commercial applications [2,3]. Load balancing and scheduling play a critical role in achieving high utilization of resources in such environments [4]. Scheduling an application is significantly complicated and challenging because of the heterogeneous nature of a grid system. Grid scheduling is defined as the process of making scheduling decisions involving allocating jobs to resources over multiple administrative domains [5]. Most of the scheduling strategies aim at reducing the *makespan* or the maximum completion time of the task which is defined as the difference between the time the job was submitted to a computational resource and the time it completed. *Makespan* also includes the time taken to transfer the data to the point of computation if that is allowed by the scheduling strategy [5].

On the other hand, in many data intensive grid applications, data can be decomposed into multiple independent sub-datasets and distributed for parallel execution and analysis. High Energy Physics (HEP) experiments fall into this category [6]. HEP data are characterized by independent events, and therefore this characteristic can be exploited when parallelizing the analysis of data across multiple sites. The DLT paradigm [7] has emerged as a powerful tool for modeling data-intensive computational problems incorporating communication and computation issues [8]. An example of this direction is the work by [6] where the DLT is used to model the grid scheduling problem involving multiple sources to multiple sinks. In that model, they did not consider the communication time. However, the scheduling in grid applications must consider communication and computation simultaneously in order to achieve high performance.

Relevant materials to the problem addressed in this paper are in [8–11], where the CDLT, ADLT and A^2DLT models are proposed. These models are proposed for scheduling divisible load data-intensive grid applications. In the CDLT model, the scheduler targets an application model wherein a large dataset is split into multiple smaller datasets [8]. Then, these datasets are processed in parallel on multiple virtual sites, where a virtual site is considered to be a collection of computing resources and data servers. However, in CDLT, the communication time for transferring load is not considered. In addition, ADLT and A^2DLT models are proposed in considering communication time as well as computation time in [10,11], respectively. These two models provide step-wise scheduling models which will be used in this paper to generate the initial solution of the IDLT model.

Our objective is to design a load distribution model by taking into account the communication time and the computation time in such a way that the entire processing time of the load is minimized. The main contributions of this paper are the closed form solutions for the minimum completion time; in addition the optimal data allocation for each processing node is

* Corresponding author.
E-mail addresses: monir_alqubati@yahoo.com (M. Abdullah), mothman@fsktm.upm.edu.my (M. Othman), hamidah@fsktm.upm.edu.my (H. Ibrahim), shamala@fsktm.upm.edu.my (S. Subramaniam).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.04.003

Table 1
Notations and definitions.

M	The total number of nodes in the system
L	The loads in the data file
α_i	The fraction of the load that the node i will receive from the data file
L_i	The amount of load that the node i will receive from the data file
w_j	The inverse of the computing speed of the node i
Z_in_i	The link between the node i and the data source
Z_out_i	The link between the node i and the aggregator
T_i	The processing time in the node i

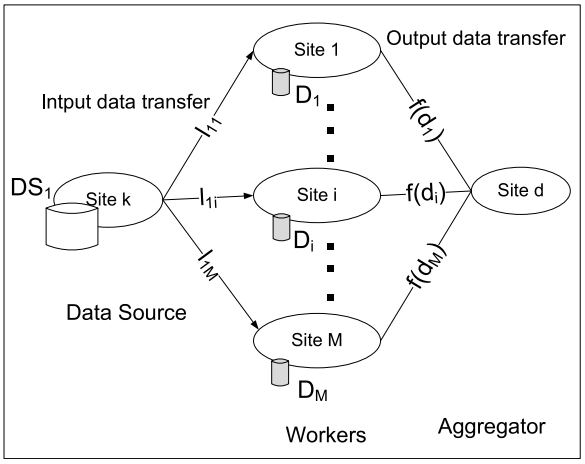

Fig. 1. Data decomposition and their processing.

obtained. We validate the model through mathematical proof and comprehensive simulations.

This rest of this paper is organized as follows: Section 2 gives the outline of the scheduling model. In Section 3, we give a detailed description of the proposed IDLT model. Section 4 presents an experimental evaluation to validate the proposed model. Finally, we summarize the findings and conclude the paper in Section 5.

2. Scheduling model

We consider the problem of scheduling large-volume loads (divisible loads) within multiple sites. Communication is assumed to be predominant between such cluster nodes and is assumed to be negligible within a cluster node [12,8,13]. This section describes the scheduling model, the notations, the cost model, and the optimality criterion that are used in our research.

We use the scheduling model that was used by [12,8,13]. It can be described as follows. 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 [8]. Let us consider job decomposition by decomposing input data objects into multiple smaller data objects of arbitrary size and processing them on multiple virtual sites. For example in theory, the High Energy Physic (HEP) jobs are arbitrarily divisible at event granularity and intermediate data product processing granularity [1]. Assume that a job requires a very large logical input data set D of a particular site. Fig. 1 shows how D is decomposed onto networks and their computing resources.

The scheduling problem is to decompose logical data into datasets across M virtual sites in a Virtual Organization (VO) given its initial physical decomposition. We assume that the divisible data can be analyzed at any site. The notations, cost model, and optimality criterion are discussed in Sections 2.1–2.3 respectively.

2.1. Notations and definitions

The notations and definitions that are 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 J can be respectively expressed as:

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

and

$$T_{Turn\_Around\_Time} = \max_{i=1}^{N} T_i.$$

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 respectively as:

$$T_{input\_cm}(i) = L_i \cdot \frac{1}{Z_i}$$

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

and

$$T_{output\_cm}(i, d) = f(L_i) \cdot Z_{id}.$$

Where, $L_i = L \cdot \alpha_i$ and the function $f(d_i)$ is 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 subtasks.

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 α_i which minimizes the turn around time of a job. The proposed model uses this cost model when evaluating solutions at each generation.

2.3. Optimality criterion

In all the literature related to the divisible load scheduling domain so far, an optimality criterion [14] is used to derive an optimal solution as follows. It states that in order to obtain an optimal processing time, it is necessary and sufficient that all the sites that participate in the computation must stop at the same time. Otherwise, load could be redistributed to improve the processing time. The timing diagram for this distributed system in the optimal case is depicted in Fig. 2. In this timing diagram, communication time appears above the axis and computation time appears below the axis.

3. Proposed IDLT model

The load 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. This model includes two steps:

3.1. Initial solution

The proposed model starts from a good initial solution. ADLT and A^2DLT models are used for this purpose. The good solution

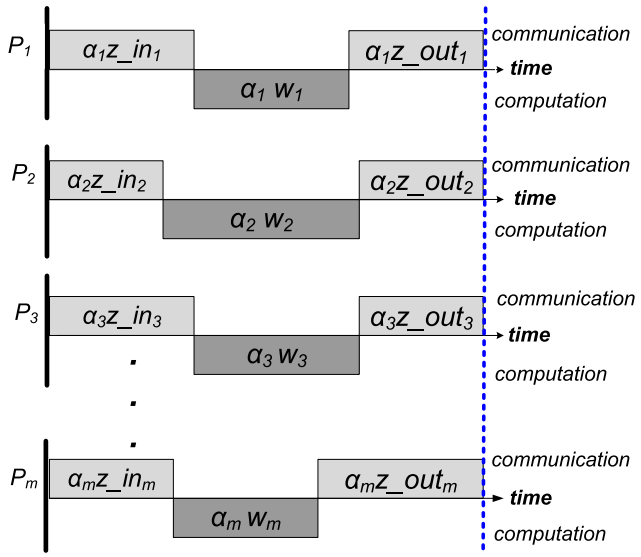

Fig. 2. Communication and computation of the sources within the system in the optimal case.

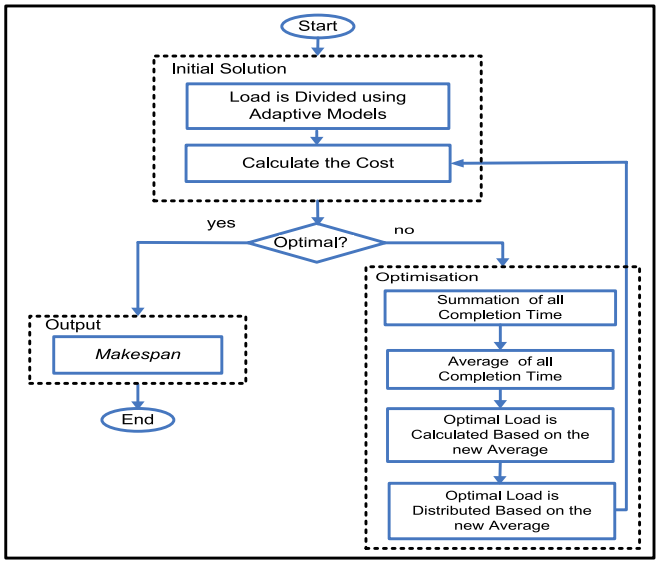

Fig. 3. IDLT framework for single source scheduling.

(minimum *makespan*) is considered as an initial solution of the iterative model. As is explained in [9,11], the ADLT model produced good results for computation-intensive applications while A^{2}DLT produced good results for communication-intensive applications.

3.2. The iterative model

The optimality criterion discussed in Section 2.3 was used in the design of our load distribution strategy. Our IDLT model involves the following steps:

1. First, the load is divided using one of the adaptive DLT models. ADLT or A^{2}DLT models will be used.
2. Then, the *makespan* is calculated using the cost model.
3. If all nodes finish at the same time, go to step 8; else go to step 4.
4. The summation (*Sum*) of the processing time of the nodes is calculated.
5. Next, the average time is calculated by $avg = Sum/M$.
6. After that, the load fraction is calculated and redistributed depending on the average (avg) based on the iterative numerical equations discussed later in this section.
7. Go to step 2.
8. The current time is the final *makespan*.
9. End.

The framework of the IDLT model for single source scheduling is shown in Fig. 3.

Here, we will discuss step by step the derivation of a closed form equation by which one can calculate the optimal fraction of the load that has to be assigned to each processing node in order to achieve the minimum *makespan* and the optimal data allocation for each processor. The processing nodes (w_i), communication links (Z_i) and applications types ($ccRatio$) are assumed to have different values.

After the *makespan* is calculated as an initial solution to the IDLT model, the $\sum_{i=1}^{M} T_i$ is computed where M is the number of processing nodes. Based on the cost model we get:

$$T_i = L_i \cdot z\_in_i + w_i \cdot L_i \cdot ccRatio + L_i \cdot z\_out_i \tag{1}$$

where z_in_i and z_out_i is the input communication time and the output communication time of link i, respectively. Then, the average of the completion time is calculated as:

$$avg = \frac{\sum_{i=1}^{M} T_i}{M}. \tag{2}$$

Now, in any iteration, the *makespan* of any nodes must be equal to avg in order to get the optimal solution. It means that the load is distributed among the processing node M equally.

While the average time is the processing time of load α, which is calculated by Eq. (1), we can recalculate the value of α if we have the avg. In general, having avg is:

$$avg = \alpha \cdot z + \alpha \cdot w. \tag{3}$$

The load fraction α can be calculated by the Eq. (4):

$$\alpha = \frac{avg}{z + w}. \tag{4}$$

In our case, for any iteration, after we calculate the time average of processing any load, we can calculate the amount of this load. Furthermore, the processing time of the first node to process α_1 is:

$$avg = \alpha_1 \cdot z\_in_1 + \alpha_1 \cdot w_1 \cdot ccRatio + \alpha_1 \cdot z\_out_1. \tag{5}$$

The α_1 is calculated by:

$$\alpha_1 = \frac{avg}{z\_in_1 + (w_1 \cdot ccRatio) + z\_out_1}. \tag{6}$$

Consequently, the α_2 of the second node is calculated by:

$$\alpha_2 = \frac{avg}{z\_in_2 + (w_2 \cdot ccRatio) + z\_out_2}. \tag{7}$$

Finally, the α_M of the Mth node is

$$\alpha_M = \frac{avg}{z\_in_M + (w_M \cdot ccRatio) + z\_out_M}. \tag{8}$$

Eq. (8) is corrected only in the last iteration (when we get the optimal solution). But in the first iteration, the last node will take the rest of the load only as:

$$\alpha_M = (1 - (\alpha_1 + \alpha_2 + \cdots + \alpha_{M-1})) \cdot L. \tag{9}$$

There are $M - 1$ equations. An additional equation called a normalization equation states that the summation of all the allocation fractions should be 1.

$$\alpha_1 + \alpha_2 + \cdots + \alpha_M = 1. \tag{10}$$

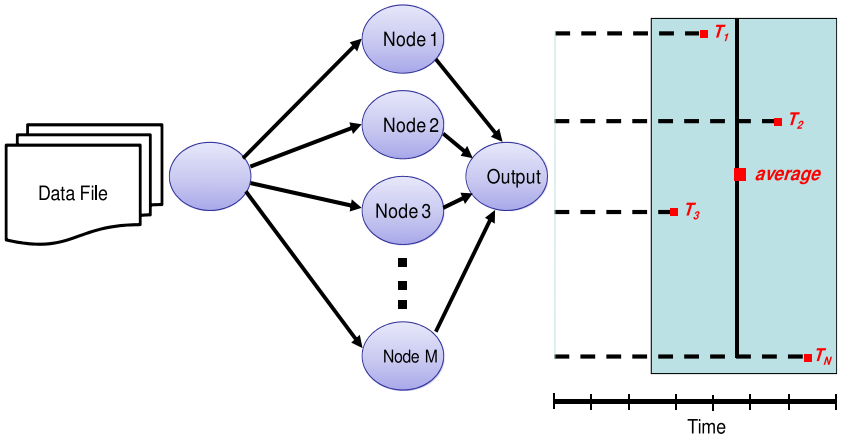

Fig. 4. Simple idea of the IDLT model for M processing nodes.

The load of the last node is not equal to the load that produces the avg time. That means that the last node still does not take the optimal load. Here, we will compute the new *makespan* (the *makespan* is calculated by the cost model that is discussed in Section 2.2 for every iteration).

Consequently, these steps are carried out in the second iteration. In this iteration, the *makespan* is reduced but still not optimal. Therefore, these steps are carried out until certain termination condition is present. The termination condition here is the optimality criterion. All nodes must finish the load processing at the same time.

The IDLT model for single source produces almost the optimal solution after some iterations. There is a very small different among the nodes processing time (small fractions). The IDLT model is as shown in Fig. 4.

In this model, all nodes will take the new load based on the new average. It means that all nodes will finish at the same time. For the last node we will take the rest of the load without considering the average. For the next iteration, the new average will be reduced. The last node will take more load than previous iteration. After some iterations the last node will finish as the same time as the others.

4. Experimental results

All the simulation studies are based on the test bed used during a large scale production effort for the (HEP experiment) CMS [2]. Large scale and unstable systems characterise the grid systems. Large scale means: (i) high numbers of data sources (e.g. database, xml files), users, and computing resources (CPU, memory, network and I/O bandwidth) which are heterogeneous and autonomous; (ii) the network bandwidth presents, on average, a low bandwidth and strong latency; and (iii) a huge volume of data [15]. In CERN, every site has a computing element and initially empty storage capacity. The processing nodes immediately start computing the load fractions once they start receiving them. It is assumed that each processing node has adequate memory/buffer space to accommodate and process all the loads received from all sources. It is also assumed that the decomposed data can be analyzed at any site by applying the same data parallel operation.

To measure the performance of the proposed model, randomly generated experimental configurations were used as in previous research [6,8,13]. The estimated expected execution times for processing a unit dataset on each site, computation speed to communication speed (r_{cb}), the network bandwidth between sites, input data size, and the ratio of output data size to input data size ($oiRatio$) are randomly generated with uniform probability over predefined ranges as follows.

Table 2
Experimental parameters and their range of values.

Parameter	Range of values
Number of nodes (M)	1–100
Data file size	1GB–1TB
Ratio r_{cb}	10–500
Computation time for 1 MB (w)	$1/r_{cb}$–$10/r_{cb}$ s
Communication speed (Z)	1–10 Mbps
Application type ($ccRatio$)	0.001 and 1000
$oiRatio$	0 and 1

- The overall performance of each model is examined by running them under 100 randomly generated grid configurations.
- The network bandwidth between sites is uniformly distributed between 1 Mbps and 10 Mbps.
- The physical data source size is randomly selected with a uniform distribution in the range of one gigabyte (GB) to one terabyte (TB). That means the data file size varies from one GB to one TB.
- It is assumed that the computing time spent in a site i to process a unit dataset of size 1 MB 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. The value of r_{cb} is varies between 10 and 500.
- The application type ($ccRatio$) varies between 0.001 and 1000. High $ccRatio$ means the application needs more communication and low computation time. The application needs less communication time and high computation time.
- The number of processing nodes M is from 1 to 100, while the number of data sources N is from 1 to 100.
- The ratio of output data size to input data size ($oiRatio$) varies from 0 to 1.

The experimental parameters and their range of values are shown in Table 2.

We examined the overall performance of each model by running them under 100 randomly generated Grid configurations. Thus, from these series of test results, it can be concluded that the IDLT gives better performance. In fact, our proposed IDLT model gives the optimal solution. All nodes stop at the same time.

As an example, we consider the system and loads with: $M = 5$ (processing nodes), $ccRatio = 0.001$, respectively. Using the IDLT model, we have the result that is clearly demonstrated in Table 3 and Fig. 6.

In the initial solution, node 1 produces the *makespan* (the maximum completion time) which is equal to 5643.84 s. This time

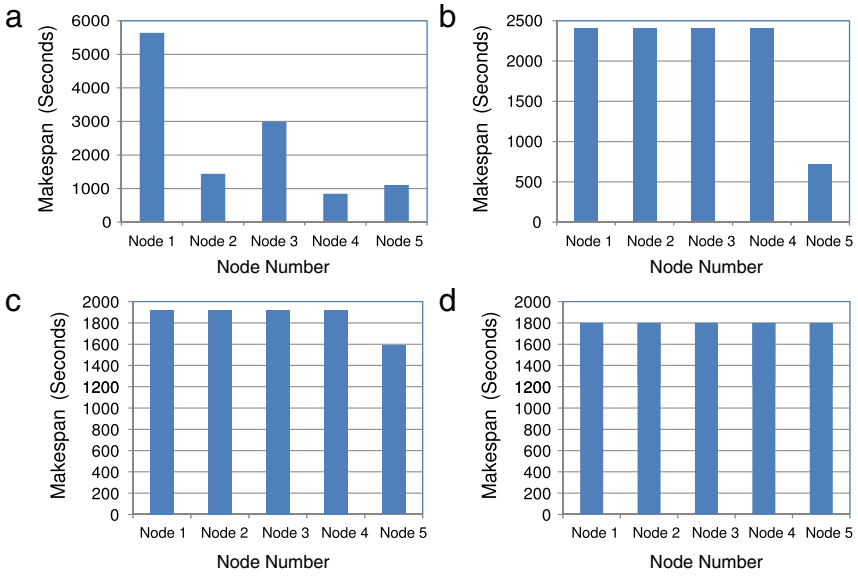

Fig. 5. Comparison of various iterations for the IDLT model (a) initial step, (b) and (c) intermediate steps, and (d) final step.

Table 3
Results of the IDLT model, step by step.

Step	Node 1	Node 2	Node 3	Node 4	Node 5
0	5643.84	1440.41	2993.02	843.48	1108.35
1	2405.82	2405.82	2405.82	2405.82	715.72
2	2067.80	2067.80	2067.80	2067.80	1319.64
3	1918.17	1918.17	1918.17	1918.17	1586.98
4	1851.93	1851.93	1851.93	1851.93	1705.32
5	1822.61	1822.61	1822.61	1822.61	1757.71
6	1809.63	1809.63	1809.63	1809.63	1780.90
7	1803.88	1803.88	1803.88	1803.88	1791.16
8	1801.34	1801.34	1801.34	1801.34	1795.71
9	1800.21	1800.21	1800.21	1800.21	1797.72
10	1799.71	1799.71	1799.71	1799.71	1798.61
11	1799.49	1799.49	1799.49	1799.49	1799.00

is reduced after one iteration to 2405.82 s and the solution is improved by 57%. Then, after several iterations, the *makespan* is reduced further to be 1799.71 s. In this example, the solution is improved by 68%. As we see in the last step, all nodes finish the processing at the same time (at 1799.00 s). Furthermore, when we increase the number of iterations, all nodes are completed at almost the same time.

Fig. 5 showed that the five processing nodes finish at almost the same time. It means that, in the last iteration, the load is distributed to the processing node almost equally.

4.1. Effect of application type

To show how these models perform on different types of applications (different *ccRatio*), Fig. 6 is created as.

In Fig. 6, the *makespan* for the CDLT, ADLT, A^{2}DLT and IDLT models is plotted against application type (*ccRatio*). The value of *ccRatio* is fixed at 1000 and the value of the number of nodes M is fixed to be 100. It can be shown from the figure that the IDLT model is the best for any type of application, as expected, because the IDLT model produces almost the optimal solution for scheduling load that is produced from a single source.

4.2. Effect of data file size

In this section, results of the *makespan* against data file size is presented. Different sizes of data files are used assuming large scale

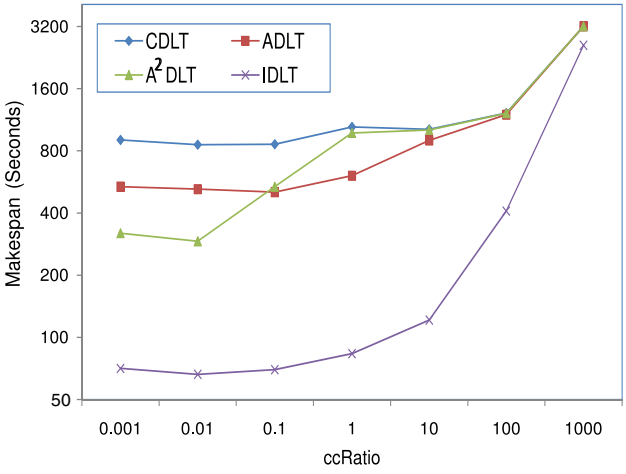

Fig. 6. *Makespan* vs. *ccRatio* for CDLT, ADLT, A^{2}DLT, and IDLT ($M = 100$).

data grids. It is varied from 1 GB to 1 TB. Fig. 7 clearly showed that the IDLT model produces better results for all sizes. Although the CDLT, ADLT and A^{2}DLT models produce the same results when the *ccRatio* is equal to 1000, the IDLT model produces a lower result due to the effectiveness of the iterative models. That is because the IDLT model produces optimal results for single source scheduling.

4.3. Effect of the number of nodes

The proposed model as well as the previous models are implemented with a variable number of processing nodes. The results are taken while the value of *ccRatio* is varied from 0.001 to 1000. Fig. 8 clearly demonstrated the performance of the four models. We can see that when the the number of nodes increases, the *makespan* decreases. Besides, when the *ccRatio* $=$ 1000, the IDLT model produced the best solution while the other models have same solution as before.

Under all criteria, we observe that the IDLT model yields the highest efficiency for any number of processing nodes and any types of applications. In contrast, the worst performance is obtained from the CDLT model. But when the number of nodes is more than around 30, the CDLT model is better than ADLT. This is

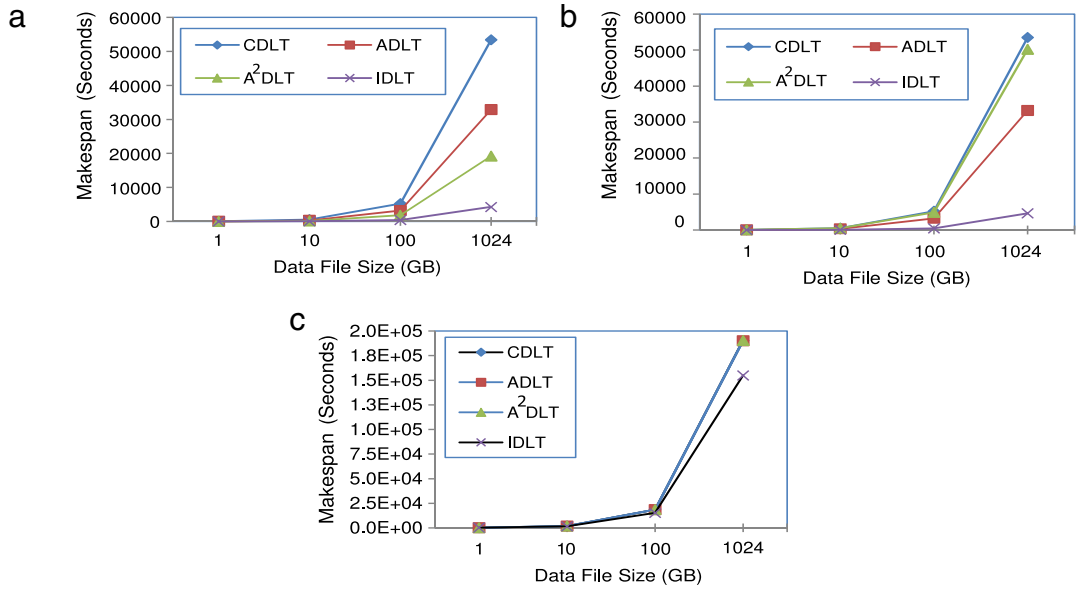

Fig. 7. *Makespan* vs. data file size for CDLT, ADLT, A^{2}DLT and IDLT($M = 100$) (a) *ccRatio* = 0.001 (b) *ccRatio* = 1 and (c) *ccRatio* = 1000.

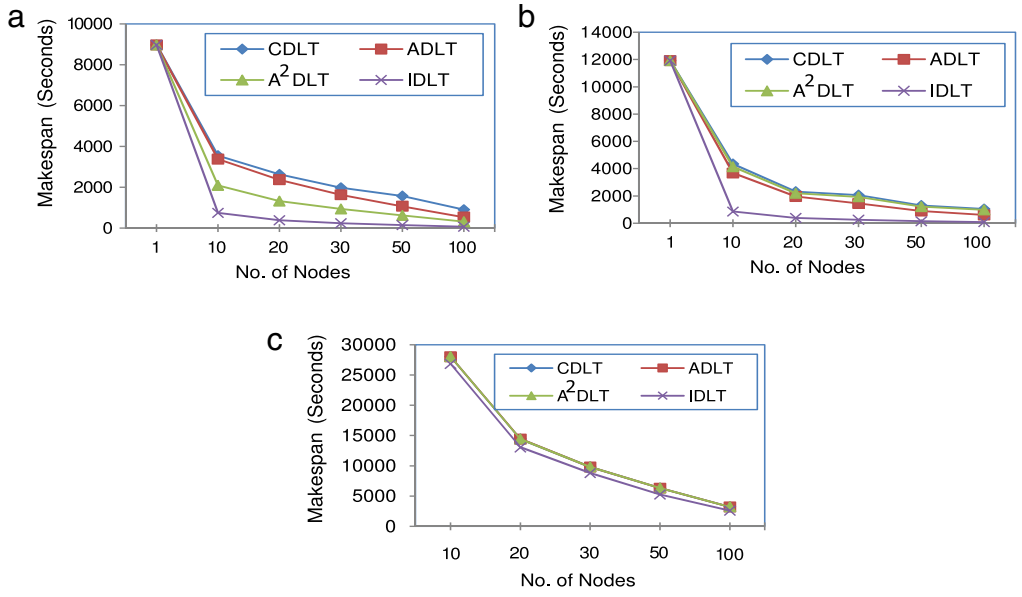

Fig. 8. *Makespan* vs. number of nodes for CDLT, ADLT, A^{2}DLT, and IDLT ($M = 100$) (a) *ccRatio* = 0.001, (b) *ccRatio* = 1, and (c) *ccRatio* = 1000.

Table 4
Percentage *makespan* improvements of IDLT against CDLT, ADLT, and A^{2}DLT models for single source.

ccRatio	*oiRatio* > 0			*oiRatio* = 0		
	CDLT (%)	ADLT (%)	A^{2}DLT (%)	CDLT (%)	ADLT (%)	A^{2}DLT (%)
0.001	92.16	86.69	77.82	90.48	77.81	1.07
1	92.01	86.27	91.46	89.05	76.42	88.24
1000	19.23	19.12	19.23	12.38	12.24	12.38
Average	67.80	64.06	62.84	63.97	55.49	33.90

when *ccRatio* = 0.001. Lastly, when *ccRatio* = 1000, all models produce the same results while the IDLT model produces the best.

4.4. Effect of ratio of output data to input data

The impact of the ratio of output data size to input data size (*oiRatio*) is shown in Fig. 9. The IDLT model performs well for any type of application, especially for applications in which *oiRatio* >0.5.

The percentage *makespan* improvements of the IDLT model against the three models is clearly shown in Table 4.

From Table 4, it is observed that the IDLT model is the best for all type of applications. That is because the IDLT model produces almost the optimal solution.

From the experimental results, it is evident that IDLT outperforms all previous models. The performance of IDLT and the other models was compared in different data grid applications (different *ccRatio*). Furthermore, when the IDLT model applies effectiveness closed form solutions, it produces an almost optimal *makespan*.

4.5. IDLT model convergence

This section discusses the convergence of the IDLT model. The convergence metric records how the initial *makespan* value is

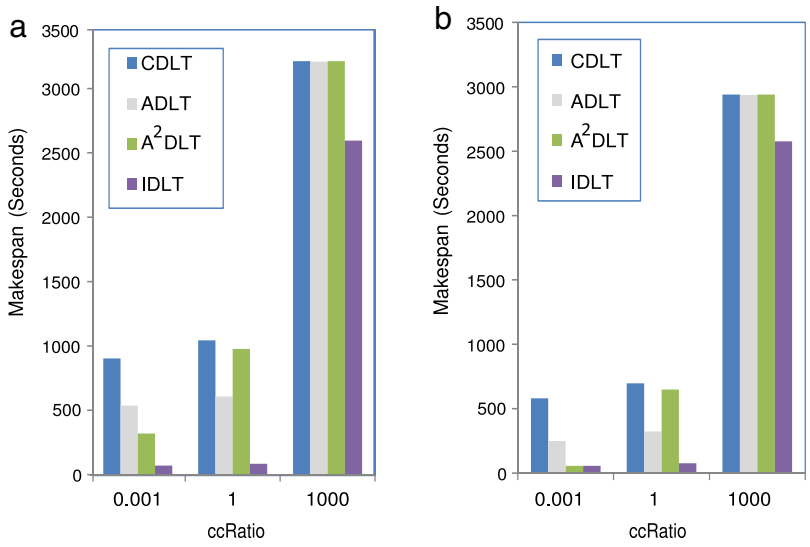

Fig. 9. The impact of output data size to input data size (*oiRatio*) (a) *oiRatio* > 0.5 and (b) *oiRatio* = 0: No output or small size of output.

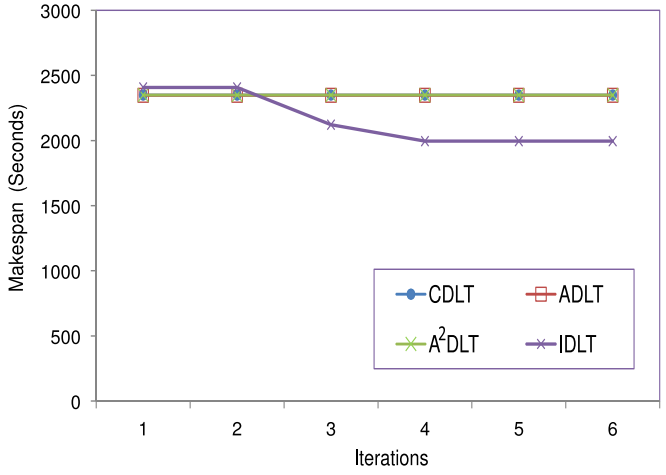

Fig. 10. Convergence of the IDLT model for single source (*ccRatio* = 1000).

minimized during the loop between the initial solution and optimal one. Fig. 10 depicts the convergence of the models for the average out of eleven executions when the *ccRatio* is equal to 1000.

When the *ccRatio* is equal to 1000, the result is optimized more by the IDLT model. As we see, all other models have the same result, whereas the result of the IDLT model is significantly reduced due to the iterative technique.

5. Conclusion

In this paper, we have developed an effective iterative model for optimal workload allocation. The IDLT model is proposed for load allocation to processors and links for scheduling divisible data grid applications. Experimental results show that the proposed IDLT model is capable of producing an almost optimal solution for single source scheduling. Hence, the proposed model can balance the processing loads efficiently. We are planning to adapt the proposed model for implementation in multiple sources.

With such improvements, the proposed model can be integrated in the existing grid middleware in order to improve their performance. It can be integrated in gLite in the workload management services to distribute the load equally.

References

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

[2] K. Holtman, J. Amundson, P. Avery, S. Aziz, L.A.T. Bauerdick, J. Branson, J.J. Bunn, P. Capiluppi, R. Clare, A. Dominici, F. Donno, I. Fisk, I. Gaines, G. Graham, C. Grandi, CMS requirements for the grid, in: Proceeding of the International Conference on Computing in High Energy and Nuclear Physics, Science Press, Beijing China, 2001.

[3] B. Tierney, W. Johnston, J. Lee, M. Thompson, A data intensive distributed computing architecture for grid applications, Future Generation Computer Systems 16 (5) (2000) 473–481.

[4] Q. Xiao, Design and analysis of a load balancing strategy in data grids, Future Generation Computer Systems 16 (23) (2007) 132–137.

[5] S. Venugopal, R. Buyya, K. Ramamohanarao, A taxonomy of data grids for distributed data sharing, management and processing, ACM Computing Surveys 38 (1) (2006) 1–53.

[6] H.M. Wong, B. Veeravalli, Y. Dantong, T.G. Robertazzi, 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, pp. 7–11.

[7] Y.C. Cheng, T.G. Robertazzi, Distributed computation with communication delays, IEEE Transactions on Aerospace and Electronic Systems 22 (1988) 60–79.

[8] S. Kim, J.B. Weissman, A genetic algorithm based approach for scheduling decomposable data grid appklications. in: IEEE Proceeding of the International Conference on Parallel Processing, Washington, DC, USA 1, 2004, pp. 406–413.

[9] A. Abraham, R. Buyya, B. Nath, Nature's heuristics for scheduling jobs on computational grids, in: Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, ADCOM, 2000, pp. 45–52.

[10] M. Othman, M. Abdullah, H. Ibrahim, S. Subramaniam, Adaptive divisible load model for scheduling data-intensive grid applications, in: Computational Science, in: Lecture Notes in Computer Science, vol. 4487, Springer Verlag, 2007, pp. 446–453.

[11] M. Othman, M. Abdullah, H. Ibrahim, S. Subramaniam, A^{2}DLT: divisible load balancing model for scheduling communication-intensive grid applications, in: Computational Science, in: Lecture Notes in Computer Science, vol. 5101, Springer Verlag, 2008, pp. 498–507.

[12] M. Tang, B.-S. Lee, X. Tang, C.-K. Yeo, The impact of data replication on job scheduling performance in the data grid, Future Generation Computer Systems 22 (3) (2006) 254–268.

[13] S. Viswanathan, B. Veeravalli, T.G. Robertazzi, Resource-aware distributed scheduling strategies for large-scale computational cluster/grid systems, IEEE Transaction of Parallel and Distributed Systems 18 (10) (2007) 1450–1461.

[14] V. Bharadwaj, D. Ghose, T. Robertazzi, Divisible load theory: a new paradigm for load scheduling in distributed systems, Cluster Computing 6 (1) (2003) 7–17.

[15] A. Hameurlain, F. Morvan, M. Samad, Large scale data management in grid systems: a survey, in: 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008, pp. 1–6.

Monir Abdullah obtained his B.Sc. in Computer Science from Mosul University, 2000. He received his M.Sc. and Ph.D. in Parallel and distributed computing from Universiti Putra Malaysia in 2005 and 2009 respectively. Currently he is a lecturer at the Department of Computer Science, College of Computer Science and Information Systems, Thamar University. His research interests are in grid computing, optimizations, parallel processing, computer networks and computer algorithms. He has published articles in several refereed international conferences and journals.

Mohamed Othman received his Ph.D. from the National University of Malaysia with distinction (Best Ph.D. Thesis in 2000). Now, he is a Professor in Computer Science and Deputy Dean of Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM) and prior to that he was a Deputy Director of the Information Development and Communication Center (iDEC) where he was in charge of the UMPNet network campus, the uSport Wireless Communication project, High Performance Enterprise Servers and the UPM DataCenter. Between 2002 and 2009, he received many gold and silver medal awards for University Research and Development Exhibitions and Malaysia Technologies Exhibition which is at the national level. His main research interests are in the fields of parallel and distributed algorithms, high-speed networking, network design and management (network security, wireless and traffic monitoring) and scientific computing. He is a member of the IEEE Computer Society, the Malaysian National Computer Confederation, and the Malaysian Mathematical Society. He has published articles in 120 national and international journals and more than 200 conference proceedings papers. He is also an associate researcher and coordinator of High Speed Machine at the Laboratory of Computational Science and Informatics, Institute of Mathematical Science (INSPEM), Universiti Putra Malaysia.

Hamidah Ibrahim is currently an associate professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. She obtained her Ph.D. in computer science from the University of Wales, Cardiff, UK in 1998. Her current research interests include databases (distributed, parallel, mobile, bio-medical, XML) focusing on issues related to integrity constraints checking, cache strategies, integration, access control, transaction processing, query processing and optimization, and data management in grid and knowledge-based systems.

Shamala Subramaniam received a B.S. degree in Computer Science from Universiti Putra Malaysia (UPM) in 1996, an M.S. (UPM) in 1999, and a Ph.D. (UPM) in 2002. Her research interests are Computer Networks, Simulation and Modeling, Scheduling and Real Time Systems. She has published several journal papers.