SIMULATED ANNEALING ALGORITHM FOR SCHEDULING DIVISIBLE LOAD IN LARGE SCALE DATA GRIDS

MONIR ABDULLAH, MOHAMED OTHMAN, HAMIDAH IBRAHIM AND SHAMALA SUBRAMANIAM

Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

monir_alqubati@yahoo.com and {mothman, hamidah, shamala}@fsktm.upm.edu.my

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 exploited using Divisible Load Theory (DLT). Many Scheduling approaches have been studied but there is no optimal solution. This paper proposes a novel Simulated Annealing (SA) algorithm for scheduling divisible load in large scale data grids. SA algorithm is integrated with DLT model and compared with the previous approaches. Experimental results show that the proposed model obtains better solution in term of makespan.

KEY WORDS: *Divisible load theory, Data grid, Simulated annealing.*

1. INTRODUCTION

In the last decade, Data Grids have increasingly become popular for a wide range of scientific and commercial applications [1, 2]. Load balancing and scheduling play a critical role in achieving high utilization of resources in such environments [3]. 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 [4]. Most of the scheduling strategies try to reduce the makespan or the Minimum Completion Time (MCT) of the task which is defined as the difference between the time when the job was submitted to a computational resource and the time it is completed. Makespan also includes the time taken to transfer the data to the point of computation if that is allowed by the scheduling strategy [4].

In many data intensive 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 [5]. 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 has emerged as a powerful tool for modeling data-intensive computational problems incorporating communication and computations issues. An example of this direction is the work by Wong *et al.* [2] where the DLT is applied to model the Grid scheduling problem involving multiple sources to multiple sinks. In this model, they did not consider the communication time. Whereas, grid applications must consider communication and computation time simultaneously to achieve high performance.

A very directly relevant material to the problem addressed in this paper is in [5] where CDLT and GA-based models are proposed for scheduling decomposable data grid applications. Reference [5] also stated that the scheduler targets an application model wherein a large dataset is split into multiple smaller datasets. These are then processed in parallel on multiple virtual sites, where a virtual site is considered to be a collection of computing resources and data servers.

When SA was first proposed [6] it was mostly known for its effectiveness in finding near optimal solutions for large-scale combinatorial optimization problems such as the traveling salesperson problem, buffer allocation in production lines, and chip placement problems in circuits (finding the layout of a computer chip that minimizes the total area). But recent approaches of SA [7] demonstrated that this class of optimization approaches could be considered competitive with other approaches for solving optimization problems. SA algorithm is also proposed for the dynamic scheduling of jobs to the geographically distributed computing resources in computational grid [8]. In our research, we propose SA algorithm for scheduling divisible load in large scale data grid. The main contribution of this paper is the successful integration of DLT model with SA model for scheduling divisible data grid applications. The model takes into account computation speed as well as communication speed. The model is validated through comprehensive simulations.

The remainder of this paper is organized as follows. Section 2 introduces the scheduling problem. A short discussion over the CDLT and GA-based scheduling approaches are presented in Section 3. In Section 4, we will present in detail the new SA scheduling approach. Section 5 presents an experimental evaluation to validate the proposed model. Finally, we summarize our findings and conclude the paper in the last section.

2. SCHEDULING MODEL

The target data intensive applications model can be decomposed into multiple independent sub tasks and executed in parallel across multiple sites without any interaction among sub tasks [3]. 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. For example, in theory, the High Energy Physic (HEP) jobs are arbitrarily divisible at event granularity and intermediate data product processing granularity [5].

Assume 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. Figure 1 shows how 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. We assume that the decomposed data can be analyzed on any site. For the notations, definition and cost model are discussed below in sections A and B respectively.

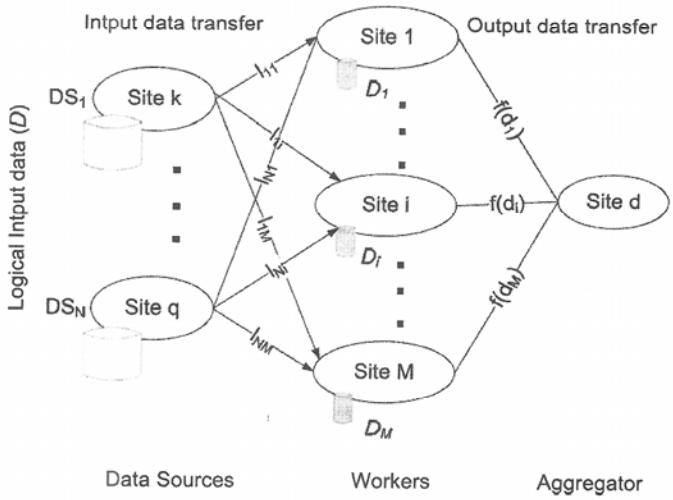

Fig. 1: Scheduling Model [3].

Notations and Definitions

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

Table 1: Terminology, Definitions, and Notations

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 i will receive from data file $_j$
L	The sum of loads in the system, where $L = \sum_{i=1}^{N} L_i$
α_{ij}	The fraction of L that node i will receive from all data file j
ω_i	The inverse of the computing speed of node i
Z_{ij}	The link between node i and data source j
T_{cp}	The computing intensity constant.
$T(i)$	The processing time in node i

Cost Model

The execution time of a subtask allocated to the site i (T_i) and the turn around time of a job J ($T_{turn\_around\_time}$) can be expressed as follows:

$$T_i = T_{input\_cm}(i) + T_{cp}(i) + T_{output\_cm}(i,d) \quad (1)$$

$$T_{turnaround\_time} = \max_{i=1,\ldots,M} T_i \quad (2)$$

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 ($T_{output\_cm(i,d)}$).

$$T_{input\_cm}(i) = \max_{k=1..N} \{l_{ki} \cdot \frac{1}{z_{ki}}\} \quad (3)$$

$$T_{cp}(i) = d_i \cdot \omega_i \quad (4)$$

$$T_{output\_cm}(i,d) = f(d_i) \cdot z_{id} \quad (5)$$

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 $\{l_{ki}\}$ which minimizes the turn-around time of a job. The proposed SA approach uses this cost model when evaluating solutions at each generation.

3. RELATED SCHEDULING APPROACHES

In this Section, we will discuss some related works done in this area of research.

Constrained DLT (CDLT)

In order to solve under-constrained system, additional constraints need to be added to this system. One possible constraint we tested, suggested in [2], is that each worker node receives the same load fraction from each data source. With this constraint, the system of equations is solvable for a unique solution. The drawbacks of this approach is that the communication time didn't consider when the load is divided.

GA-Based Scheduler

GA-based Scheduler is introduced for reducing makespan of divisible load in large scale data grids. The scheduler targets an application model wherein a large dataset is split into multiple smaller datasets and these are then processed in parallel on multiple virtual sites, where a virtual site is considered to be a collection of compute resources and data servers. In GA, the solution to the scheduling problem is represented as a chromosome in which each gene represents a task allocated to a site. Each sub gene is associated with a value that represents the fraction of a dataset assigned to the site, and the whole gene is associated with a value denoting the capability of the site given the fraction of the datasets assigned, the time taken to transfer these fractions and the

execution time. The chromosomes are mutated to form the next generation of chromosomes. At the end of iteration, the chromosomes are ranked according to makespan and the iteration stops at a predefined condition. Since the objective of the algorithm is to reduce the completion time, the iterations tend to favor those tasks in which the data is processed close to or at the point of computation, thereby exploiting the spatial locality of datasets.

In this approach, the communication time is considered when they divide the load but the drawback of this approach is the time consuming for chromosome representation and initial population.

4. SA-BASED SCHEDULING

SA model is used to approximate the solution of very large combinatorial optimization problems [6] (e.g. NP-hard problems). It is based upon the analogy between the annealing of solids and solving optimization problems. SA has been used in operations research to successfully solve a large number of optimization problems such as the Traveling Salesman problem and various scheduling problems [7]. Recently SA is proposed to be applied in grid scheduling problem in computational grid [8]. Here, it is applied to the problem of application scheduling in a data grid environment.

SA Parameters

The following fundamental terminology about SA is useful before going to a detailed description of SA.

Objective Function: An objective function maps an input vector x into a scalar $E = f(x)$, Where each x is viewed as a point in an input space. The task of SA is to sample the input space effectively to find an x that minimizes E.

Generating Function: A generating function $g(\cdot,\cdot)$ specifies the probability density function of the difference between the current point and the next point to be visited. Specifically, Δx $(= x_{new} - x)$ is a random variable with probability density function $g(\Delta x, T)$, where T is the temperature. For common SA used in combinatorial optimization applications $g(\cdot,\cdot)$ is a function independent of temperature T.

Acceptance Function: After a new point x_{new} has been evaluated, SA decides whether to accept or reject it based on the value of the acceptance function h(,). The most commonly used acceptance function is the Boltzmann distribution function

$$h(\Delta E, T) = \exp(-\Delta E / T) \quad (6)$$

Where T is the temperature, and ΔE is the energy difference between x_{new} and x.

$$\Delta E = f(x_{new}) - f(x) \quad (7)$$

The common practice is to accept x_{new} with probability $h(\Delta E, T)$. Note that when ΔE is negative, SA tends to accept the new point because it reduces the energy. When ΔE is positive, SA may accept the new point and end up in a higher energy state or it may not accept the point. Lower the temperature; the less likely SA is to accept any significant high-energy states.

Annealing Schedule: An annealing schedule regulates how rapidly the temperature T goes from high to low values. The easiest way of setting an annealing schedule is to decrease the temperature T by a certain percentage at each iteration.

Usually the move set $M(x)$ represents a set of neighboring points of the current point x in the sense that the objective function at any point of the move set will not differ too much from the objective function at x.

Initial Solution

Some experiments were carried out with random initial solutions. Computational time to converge towards a good solution was prohibitively long. Therefore, CDLT model was used as an initial solution, which yielded a good solution. From this point the best result is taken (the minimum makespan) before entering the SA algorithm. It is better to start with the best solution that has been heuristically built.

SA-Based Algorithm

The SA algorithm for the scheduling problem is shown below.

Generate initial solution by CDLT model.
Current Energy E equal the current solution
While (t_{stop}>0 and T>T_{final})
For i=0 to M do step 5 through 9.
Inversion move set
Calculate New solution .
If the newsolution <= current solution then
the solution is accepted.
Else
R=Random Number(0<R<1),
Y=exp(-ΔE/T),
If(R<Y), then
accept the solution,
Else reject it.
If we accept the solution then
Current energy equal new energy,
Else
the old solution will kept
If we accept the new solution then
tstop=ts,
Else
decrement tstop by 1 (t_{stop}= t_{stop}-1).
End for
*Change the temperature by T=T*α.*
End of while

SA with DLT Model

The new solution of SA model is calculated based on DLT model. We take the average of the old solution then we redistribute the load again considering the node computation time and communication time.

5. EXPERIMENTAL RESULTS

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 Mbyte/second and 10 Mbyte/second. 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 1 GB to 1 TB. We assume 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 *rcb* is the ratio of computation speed to communication speed.

We performed an experiment with 100 nodes and 100 data files. Also we varied the types of applications (*ccRatio*) from 0.001 to 1000 (*ccRatio* is the non-zero ratio of computation and communication). After many careful testing of many cases the number of rounds was set at 10, starting temperature at 1000, final temperature at 0.05, cooling rate at 0.9, the number of iterations per temperature at only 10, and rounds at 10. If no improvement is noticed we set the stop condition at only 10.

The GA parameters are fixed as the previous research. We fixed GA related parameters ($r_x = r_{gx} = 0.6$, $r_m = 0.5$, and $r_{gm} = 0.6$) with values that they found after some preliminary experiments. After many simulations we obtained the results in Table 2 and Fig.2.

Table 2: Makespan for the Three Models.

ccRatio	CDLT	GA	SA
0.001	29744.08	19220.05	5661.64
0.01	28249.86	18301.77	5717.96
0.1	30843.30	20182.60	5851.31
1	30172.42	20471.92	6336.51
10	32549.29	99529.40	10973.96
100	54618.89	743619.6	39163.24
1000	278504.70	6470095.00	266571.40

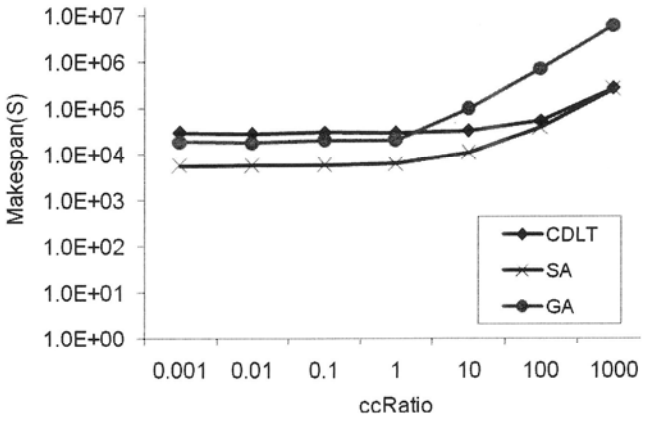

Fig. 2: Makespan of the Three Algorithms.

SA model performs better than both CDLT and GA models especially when c*cRatio* is less than 1000. When the *ccRatio* is equal to 1000, the CDLT model result is very near to the SA model results duo to the big difference between the communication time and the computation time. On the other hand, when the *ccRatio* is equal to 1000 the results depend only on the computation time.

When we compare the three models with different data file sizes, SA model also performs the best among them in terms of makespan. As shown in Fig. 2, SA and GA models performs better than CDLT model when ccRatio is equal to 1. Here, we fixed *ccRatio* at 1 and we varied data file size as 1, 10, 100 and 1024 GB. The results are shown in Table 3 and Fig. 3.

Table 2: Makespan vs. Data File Size for the Three Models (*ccRatio* = 0.001).

ccRatio	CDLT	GA	SA
1	1800.76	371.48	1154.41
10	18775.36	3748.32	13329.14
100	163106.7	36556.65	126727.5
1000	1901907	383375.2	1325034

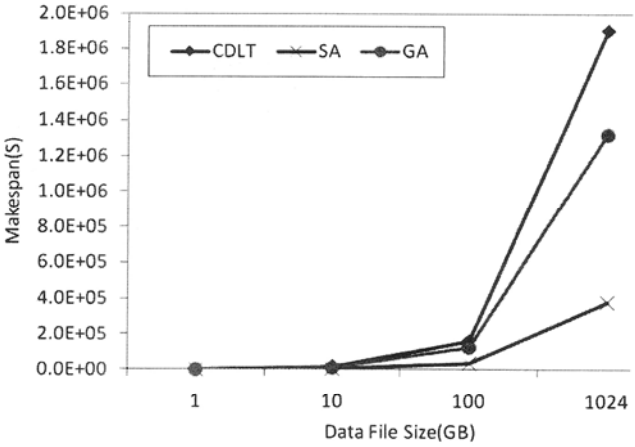

Fig. 3: Makespan vs. Data File Size for the Three Models (*ccRatio* = 0.001).

The results in Table 1 and Fig. 3 show that the SA produces better performance than the two models in terms of makespan due to the successful integration.

6. CONCLUSION AND FUTURE WORK

In this research, SA was implemented for scheduling divisible load in large scale data grids. The objective was to minimize the completion time. The results showed that SA could be used for load scheduling efficiently.

Different operators and parameters of SA algorithm were used. To compare the performance of the SA algorithm to the GA algorithm, many cases were tested and the results were analyzed. With such improvement, the proposed model can be integrated in the existing data grid schedulers in order to improve the performance.

REFERENCES

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

[2] H.M. Wong, D. Yu, V. Bharadwaj, T.G. Robertazzi., "Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints", 2003.

[3] S. Kim, J. B. Weissman: "A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications", Proceedings of the International Conference on Parallel Processing (ICPP), August 15 - 18, 2004.

[4] Venugopal S., Buyya R., Ramamohanarao, K.: "A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing", *ACM Computing Surveys*, Vol. 38(1), pp. 1-53, 2006.

[5] K. Holtman, "CMS Requirements for the Grid", in Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2001), 2001.

[6] S. Kirkpatrick. "Optimization by Simulated Annealing: Quantitative Studies". *Journal of Statistical Physics*, Vol. 34(5-6), pp. 975–986, 1984.

[7] J. S Jang,., R., C. T. Sun, and E. Mizutani, "Neuro-Fuzzy and Soft Computing in Computational Approach to Learning and Machine Intelligence. Prentice Hall, 1996.

[8] A. Abraham, R. Buyya, and B. Nath., "Nature's Heuristics for Scheduling Jobs on Computational Grids", Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, 2000.