The Branch & Bound Algorithm Improvement in Divisible Load Scheduling with Result Collection on Heterogeneous Systems by New Heuristic Function

Submitted 26th June 2011; accepted 2nd September 2011

Farzad Norouzi Fard, Sasan Mohammadi, Peyman Parvizi

Abstract:

In this paper we propose a new heuristic function for branch & bound algorithm. By this function we can increase the efficiency of branch & bound algorithm. Divisible loads represent computations which can be arbitrarily divided into parts and performed independently parallel. The scheduling problem consists in distributing the load in a heterogeneous system taking into account communication and computation times, so that the whole processing time is as short as possible. Since our scheduling problem is computationally hard, we propose a branch & bound algorithm. By simulating and comparing results it is observed which this result produces better answers than other methods, it means that branch & bound algorithm have less total average of relative error percentage in the variety Heuristic functions.

Keywords: *divisible load scheduling, heterogeneous systems, branch & bound algorithm*

1. Introduction

Divisible loads form a special class of parallelizable applications, which if given a large enough volume, can be arbitrarily partitioned into any number of independently and identically processable load fractions. Divisible load theory (DLT) is the mathematical framework that has been established to study divisible load scheduling (DLS) [1, 2]. The problem of working scheduling heterogeneous system has specific importance because of the necessity of optimize using calculating processors and also spending less time for performing of scheduling algorithms. In this paper we study divisible load scheduling with result collection on heterogeneous which has star network. In a star connected network where the center of the star acts as the master and holds the entire load to be distributed, and the points of the star form the set of slave processors, the basic principle of DLT to determine an optimal schedule is the AFS (All nodes Finish Simultaneously) policy [3]. In heterogeneous system, processors Efficiency, communication network topology and speed of network lines can be different. Scheduling works in heterogeneous system is computationally hard. One of the computation models is divisible load. Divisible load model originated in the late 1980s [4, 5]. Surveys of divisible load theory (DLT), including applications, can be found in [1, 6]. DLT proved to be a valuable tool for modeling processing of big volumes of data [7, 8] includes image processing [9], signal processing, data mining and research in Database [10]; calculate linear algebra [11] and multimedia functions [12]. Distributing the load causes inevitable communication delays. To shorten them, the load may be sent to processors in small chunks rather than in one long message. This way the computations start earlier. Such multi-installment or multi-round divisible load processing was proposed first in [13]. Memory limitations for single-installment communications were studied in [14], where a fast heuristic has been proposed. In [15] it was shown that this problem is NP-hard if a fixed startup time is required for initiation of communications. In this theory we use master-slave model. The load located on master. Master computer divides divisible load between slaves, when slave computers received all load, start processing. Each slave computers after finishing of processing report the result to master. The problem consists in finding a communication sequence, the schedule of communications from the originator to the workers, and sizes of transferred load pieces, so the total responding time becomes minimum. In previous researches amount of slave results hypothesized low so that we ignore time delay for sending this data to master; but nowadays, researchers hypothesizing time delay for returning back slave results to master computer. If M is number of computer, to consider different arrangements, time complexity is O⟦(m!)⟧^2. it has not already represented a certain algorithm with poly-nominal time complexity that can produce answer less time in all cases but existent creative ways are LifoC, FifoC [16, 17], ITERLP [18], Sport [19, 20], GA [21], and Branch & Bound LifoC. Our aim is to suggest Branch and bound algorithm for solving divisible load scheduling with result collection on heterogeneous systems. The rest of this paper is organized as follows. In section 2 the problem is formulated. Section 3 describes Branch and bound Copt algorithm for solving DLS problem. The results presented in section 4. The last section is dedicated to conclusions.

2. System model and problem definition

The network model to be considered here consists of (M + 1) processors interconnected through M links in a single-level tree fashion as shown in Fig. 1.

In this paper we assume star interconnection. A set of working processors $\{p_1, \dots, p_m\}$ is connected to a central server called master. A processor is a unit comprising a CPU, memory and a hardware network interface. The CPU and network interface can work in parallel so that simultaneous communication and computation is possible. $\{E_1, E_2, .., E_m\}$ Is the set of computation parameters of the slave computers, and $\{C_1, C_2, \dots, C_m\}$ is the set of communication parameters of the network links. E_k Is the reciprocal of the speed of processor p_k, and C_k is the

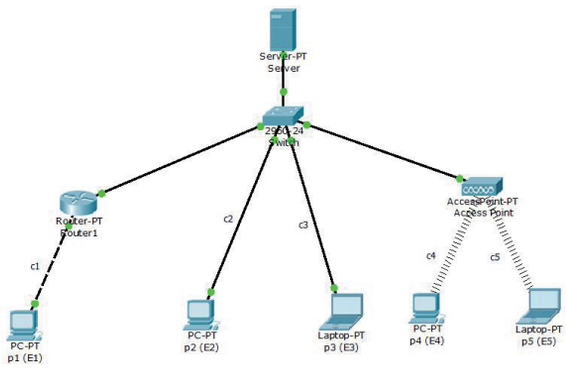

Fig. 1. A heterogeneous star network

reciprocal of the bandwidth of link l_k. In this model, L is the whole dividable load that exists in master computer. Since it does not damage problem, we suppose that L=1. The source p_0 splits L into parts and sends them to the respective processors $p_1, \ldots, p_m$ for computation. Each such set of m parts known as a load distribution $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. All processors follow a single-port and no-overlap communication model, implying that processors can communication with only one other processor at the time, and communication and computation cannot occur simultaneously. If the allocated load fraction is α_k, then the returned result is equal to $\delta\alpha_k$, where $0 \leq \delta \leq 1$. The constant δ is application specific, and is the same for all processors for a particular load L. for a load part α_k, $\alpha_k C_k$, is the transmission time from p_0 to p_k, $\alpha_k C_k$, is the time it takes to perform the requisite processing on α_k, and $\delta\alpha_k C_k$ is the time it takes p_k to transmit the results back to p_0. σ_a and σ_c are two permutation of order m that represent the allocation and collection sequences respectively $\sigma_a[k]$ and $\sigma_c[k]$ denote the processor number that occurs at index $k \in \{1,..,m\}$. $\sigma_a[k]$ and $\sigma_c[k]$ are two lookup functions that return the index of the processor k in the allocation and collection sequences. Purpose of scheduling is to find the sequence pair (σ_a, σ_c), and $\alpha_{[1\ldots k]}$ that minimize total processing time. The total processing time is started from the time of load distribution until receiving the last process from master processors. Result collection phase begins only after the entire load fraction has been processed, and is ready for transmission back to the source. This is known as a block based system model, since each phase forms a block on the time line Fig. 2.

$$\sum_{j=1}^{\sigma_a(k)} \alpha_{\sigma_a[j]} C_{\sigma_a[j]} + \alpha_K E_K + \sum_{j=\sigma_c(k)}^{m} \delta\alpha_{\sigma_c[j]} C_{\sigma_c[j]} \leq T$$

$$\sum_{j=1}^{m} \alpha_{\sigma_a[j]} C_{\sigma_a[j]} + \sum_{j=1}^{m} \delta\alpha_{\sigma_c[j]} C_{\sigma_c[j]} \leq T$$

$$\sum_{j=1}^{m} \alpha_j = J$$

$$T \geq 0, \alpha_k \geq 0, \quad k = 1, \ldots, m$$

As σ_a and σ_c are determined, we can find $\alpha_{[1\ldots k]}$ with linear programming as below:

In the above formulation, for a pair (σ_a, σ_c), (1) imposes the no-overlap constraint. The single- port communication model is enforced by (2). The fact that the entire load is distributed among the processors is ensured by (3). This is known as the normalization equation. The non-negativity of the decision variables is ensured by constraint (4) [22]. By using branch and bound algorithm to find $\sigma_{a[1\ldots m]}$, $\sigma_{c[1\ldots m]}$ and $\alpha_{[1\ldots k]}$. There is (m!) Possible permutations each of σ_a and σ_c, and the linear program has to be evaluated $(m!)^2$ times to determine the globally optimal solution.

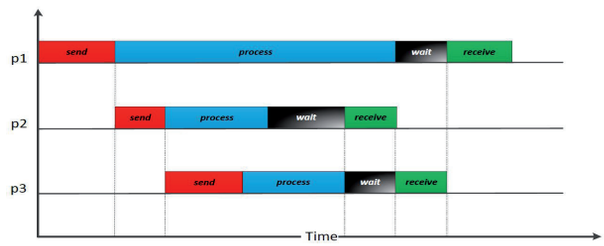

Fig. 2. Schedule for M=3

3. Branch and bound algorithm for solving DLS problem

Branch and bound algorithm is one of the trees and graphs traversal and exploring methods. Branch and bound algorithm is performed like below:

- Tree traversal,
- heuristic function,
- pruning branches.

At the beginning the root node is selected, once the root is selected its children will be created. After that heuristic function will work on all children and compare their answers. Then it will select the child who had the best result and it repeats this action until the result is found. We probably can find many answers for DLT, bout Branch and bound algorithm ended when the first answer is found. Branch and bound algorithm Travers tree as BFS and use heuristic functions for pruning branches. In Fig. 3 we display how to extend nodes.

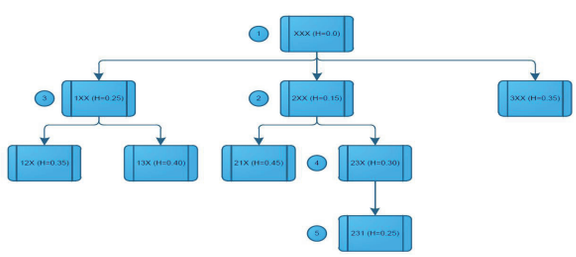

Fig. 3. Extending node in branch and bound

In our tendered algorithm (Branch & Bound Copt), first the selected processor and its father are located in allocations list, then total slaves are located in allocation list by the best C with E between them, after that we call heuristic function with this data.

4. Computational experiments

In experiments, we compared efficiency of Branch & Bound Copt algorithm by Branch & Bound LifoC, Sport, LifoC and Genetic Algorithms. We performed our Tests

by Amd Athelon Dual 3.0 Ghz with 2 Gigabyte RAM in Matlab environment. To display a heterogeneous system we consider 25 different cases of C and E. For every 25 cases, m value of C and E produced randomly. In all tests, we calculated time of process for each algorithm. If T_{opt} shows us the time of process for optimal algorithm and T_v shows the time of process for other algorithms, the percentage of relative error (ΔT_v) was calculated as formulation (5).

$$\Delta T_v = \frac{T_v - T_{opt}}{T_{opt}} \times 100\% \quad (5)$$

Since we produce 25 different cases of heterogeneous system, the average of relative error percentage is calculating as formulation (6).

$$\overline{\Delta} T_v = \frac{\sum_{k=1}^{25} (\Delta T_v)_k}{25} \quad (6)$$

In order to consider the effects of & parameter in mention algorithm, the result time obtains experiments which have been done for M=4,5 and δ=0.1,0.2,...1,

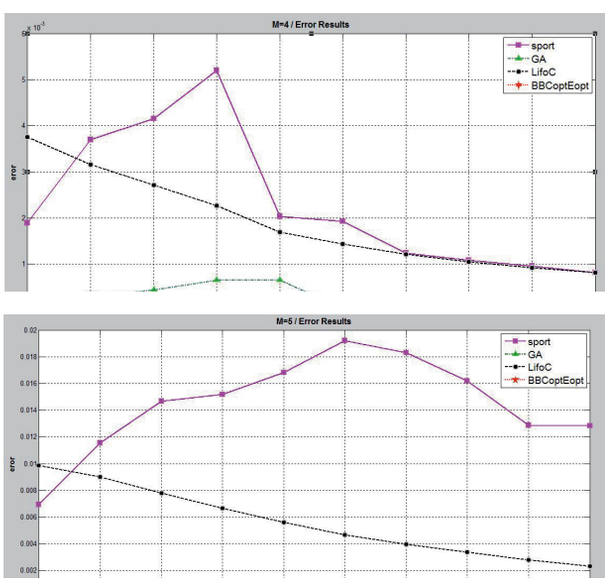

Fig. 4. Average of relative error percent for m=4,5

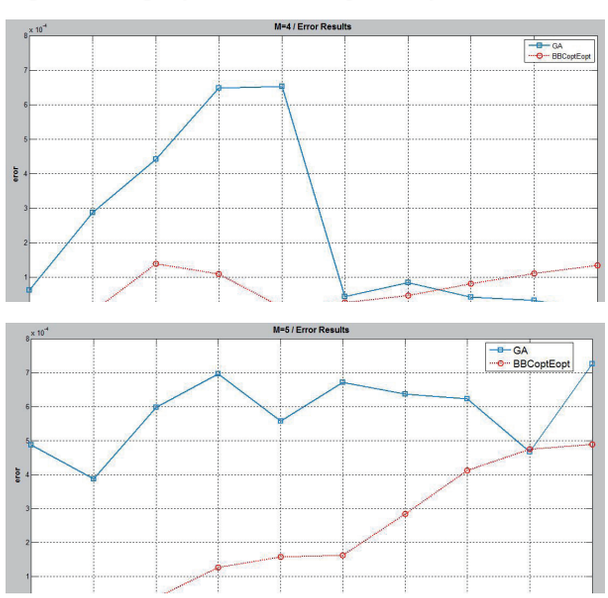

Fig. 5. Average of relative error percent for m = 4,5

Table. 1. Run time & average of relative error percentage for m=5 & δ=0.7

Algorithm	Run time	Average of relative error percentage
Optimal algorithm	182.6719	0
Branch & Bound Copt algorithm	0.2125	0.000283476
Branch & Bound LifoC algorithm	0.2	0.000299117
Genetic algorithm	30.5712	0.000637334
LifoC algorithm	0.0125	0.0039602808
FifoC algorithm	0.015	0.074704891
Sport algorithm	0.0025	0.183.05

and the average of relative errors percentage has been shown in Fig (4, 5). In these figs, we see average error percentage of Genetic algorithm, Sport, LifoC, Branch and Bound LifoC and Branch & Bound Copt for 4 and 5 slave computers.

As displayed in Fig. 4, when we have 4 slaves computer, Branch & Bound Copt algorithm in much δ value has lowest average of relative error percentage. Considering the running time being less in Branch and Bound algorithm, we can introduced it as the best algorithm. With respect to the efficiency of Branch & Bound Copt algorithm, Branch & Bound LifoC algorithm and Genetic algorithm rather than the other two, we compare them in Fig. 5.

For m=5 and δ=0.7, The Run time& average of relative errors percentage for all of algorithm has been shown in Table 1.

5. Conclusion

In this paper, a new heuristic algorithm, Branch and Bound Copt, for the scheduling of divisible loads on heterogeneous systems and considering the Result collection phase is presented. A large number of simulations are performed and it is found that Branch and Bound Copt consistently delivers near optimal performance.

As future work, an algorithm with similar performance, but with better cost characteristics than Branch and Bound Copt needs to be found. Another important area would be to extend the results to multi-level processor trees.

AUTHORS

Farzad Norouzi Fard* – Mechatronic Department Islamic Azad University – South Tehran Branch,Tehran, Iran, st_f_norouzi@azad.ac.ir

Sasan mohammadi – Mechatronic Department Islamic Azad University – South Tehran Branch,Tehran, Iran, s_mohammadi@azad.ac.ir

Peyman Parvizi – Mechatronic Department Islamic Azad University – South Tehran Branch,Tehran, Iran, st_p_parvizi@azad.ac.ir

References

[1] Bharadwaj V., Ghose D., Mani V., Robertazzi T.G., *Scheduling Divisible Loads in Parallel And Distributed Systems*, IEEE Computer Society Press, Los Alamitos CA, 1996.

[2] Lee C.H., Shin K.G., "Optimal task assignment in homogeneous networks", *IEEE Trans. Parallel Distrib. Syst*., vol. 8, no. 2, Feb. 1997, pp.119-129.

[3] Barlas G.D., "Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of divisible load on arbitrary processor trees", *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 5, May 1998, pp.429-441.

[4] Agrawal R., Jagadish H.V., "Partitioning Techniques for Large-Grained Parallelism", *IEEE Transactions on Computers*, vol. 37, no. 12, 1988, pp. 1627-1634.

[5] Cheng Y.-C., Robertazzi T.G., "Distributed computation with communication delay", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, 1988, pp. 700-712.

[6] Robertazzi T.G., "Ten reasons to use divisible load theory", *IEEE Computer*, vol. 36, 2003, pp. 63–68.

[7] Bharadwaj V., Ghose, D., Robertazzi, T. G., "Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems", *Cluster Computing*, vol. 6, no. 1, Jan. 2003, pp. 7-17.

[8] Jingxi J., Bharadwaj V., Ghose D., "Adaptive Load Distribution Strategies for Divisible Load Processing on Resource Unaware Multilevel Tree Networks", *IEEE Transactions on Computers*, vol. 65, no. 7, 2007, pp. 99-1005.

[9] Li X., Bharadwaj V., KO C. C., "Distributed Image Processing on a Network of Workstations", *International J. Computers and Applications*, vol. 25, no. 2, 2003, pp. 1-10.

[10] Blazewicz J., Drozdowski M., Markiewicz M., "Divisible Task Scheduling: Concept and Verification", *Parallel Computing*, vol. 25, 1990, pp. 87-98.

[11] Chan S., Bharadwaj V., Ghose D., "Large Matrix-Vector Products on Distributed Bus Networks with Communication Delays Using the Divisible Load Paradigm: Performance and Simulation", *Math. And Computers in Simulation*, vol. 58, 2001, pp.71-92.

[12] Altilar D., Paker Y., "Optimal Scheduling Algorithms for Communication Constrained Parallel Processing". In: *Proc. 8th Int'l Euro-Par Conf.*, 2002, pp. 197-206.

[13] Bharadwaj V., Ghose D., Mani V., "Multi-installment Load Distribution in Tree Networks with Delays", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, 1995, pp. 555-567.

[14] Li X., Bharadwaj V., Ko C. C., "Processing divisible loads on single-level tree networks with buffer constraints", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, 2000, pp. 1298-1308.

[15] Drozdowski M., Wolniewicz P., "Optimum divisible load scheduling on heterogeneous stars with limited memory", *European Journal of Operational Research*, vol. 172, 2006, pp. 545-559.

[16] Rosenberg A. L., "Sharing Partitionable Workloads in Heterogeneous NOWs: Greedier Is not Better", *IEEE International Conf. on Cluster Computing*, Newport Beach, CA, Oct. 2001, pp. 124-131.

[17] Beaumont O., Marchal L., Rehn V., Robert Y., "FIFO Scheduling of Divisible Loads with Return Messages Under the One Port Model". In: *Proc. Heterogeneous Computing Workshop HCW'06*, April 2006.

[18] Ghatpande A., Nakazato, H., Watanabe, H., Beaumont, O., "Divisible Load Scheduling with Result Collection on Heterogeneous Systems". In: *Proc. Heterogeneous Computing Workshop (HCP 2008)*, April 2008.

[19] Ghatpande A., Nakazato H., Beaumont O., Watanabe H., "SPORT: An Algorithm for Divisible Load Scheduling With Result Collection on Heterogeneous Systems", *IEICE Transactions on Communications*, vol. E91-B, no. 8, August 2008.

[20] Ghatpande A., Nakazato H., Beaumont O., Watanabe H., "Analysis of Divisible Load Scheduling with Result Collection on Heterogeneous Systems", *IEICE Transactions on Communications*, vol. E91-B, no. 7, July 2008.

[21] Suresh S., Mani V., Omkar S. N., Kim H. J., "Divisible Load Scheduling in Distributed Systems with Buffer Constraints: Genetic Algorithm and Linear Programming Approach", *International Journal of Parallel, Emergent and Distributed Systems*, vol. 21, no. 5, Oct. 2006, pp. 303-321.

[22] Vanderbei R. J., *Linear Programming: Foundations and Extensions*, 2nd Ed., *International Series in Operations Research & Management*, vol. 37, Kluwer Academic Publishers, 2001.