Hybrid Divisible Load Theory

H. J. Kim*, Kiseb Kim*, Yongsoo Choi*, Dal Ho Lee**
* Dept. of Control/Instrumentation Eng., Kangwon National University, Chunchon 200-701, Korea
Tel : +82-33-250-6343 Fax : +82-33-242-2059 E-mail: khj@kangwon.ac.kr
**Dept of Electronic Engineering, Kyungwon University, Sungnam 461-702, Korea
Tel : +82-31-750-5320 Fax : +81-31-758-5319 E-mail: dhlee@kyungwon.ac.kr

Abstract: New concept of hybrid divisible load theory is introduced in this paper. Hybrid system deals with a combination of modularly divisible load and arbitrarily divisible load. Main idea of hybrid divisible load theory is introduced with a simple example. A condition of optimality is derived for the hybrid case.
Keywords: Parallel and distributed system, Scheduling.

1. INTRODUCTION

Main objective in parallel and distributed computing systems is the minimization of processing time of the work/jobs by exploiting concurrent computing across multiple processors. Minimizing the processing time involves designing efficient scheduling algorithms as well. In general the scheduling problem addresses the following question: *What is the best possible way to organize a given work load so that it can be completed in the shortest possible time?* An efficient scheduling algorithm distributes the workload (or processing load) in an optimal manner to the set of available processors in the system so that the processing time of the entire load is minimum. Traditionally, there has been a large amount of research works available in the literature in the context of scheduling indivisible and modularly divisible loads (see Figure 1).

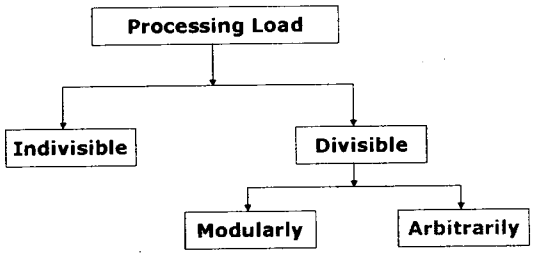

Figure 1. Scheduling schemes based on divisibility

During last decades, there is a great deal of attention focused on arbitrarily divisible load scheduling problem in a distributed computing system/network consisting of processors interconnected through communication links. Research in the scheduling of divisible loads started in 1988. A divisible load can be divided into any number of fractions, and can be processed independently on the processors, as there are no precedence relationships. In other words, divisible load has the property that all the elements in the load require the same type of processing. Thus, this load can be partitioned into any number of load fractions, and can be processed independently in the processors available in the distributed computing system. In a distributed computing environment, this load originates at one of the processors. This processor divides the load into many fractions, keeps one of the fractions for itself to process/compute and sends the remaining load fractions to other processors in the network. The objective here in this scheduling problem is that of finding the load fractions assigned to each processor in the network so that the processing time of the entire processing load is as small as possible and, if possible, minimum. The load fractions assigned to the processors are processed in parallel. In a distributed computing system, the communication delay (i.e., the time to send a load fraction to a processor) plays a major role in determining the load fractions assigned to the processors that minimizes the processing time of the entire load.

The original problem of scheduling divisible loads incorporating the communication delay is addressed in the context of distributed intelligent sensor networks in [6]. In [6], the timing diagram representation of the load distribution process and the recursive load distribution equations are presented. This problem is framed as a partitioning technique for large grained parallelism and a linear programming formulation is given in [1]. The methodology from [6] is extended to single level tree network and bus network in [2, 7]. In these studies [2, 6, 7], the load fractions assigned to the processors are obtained by assuming that all the processors involved in the computation process stop the computing at the same time instant. This assumption has been shown to be necessary and sufficient condition to obtain optimal processing time in a linear network [14]. Using the concept of processor equivalence, an analytical proof for optimal load sharing in a bus network is discussed in [15]. However, it has been rigorously proved that this condition is true only in a restricted sense [4] (i.e., for the case of a heterogeneous single level tree network). A closed-form expression for the processing time is presented in [3, 12] for the case of a single level tree network, and using this closed-form expression, optimal sequence of load distribution and optimal arrangement of links and processors in the network are obtained in [3]. For the case of a homogeneous network (all the processors have the same computational capability and all the links have the same communication capability in the network) an asymptotic performance analysis for the processing time is carried out in [5, 11, 13]. The

domain of divisible load scheduling has simulated considerable amount of interest among researchers and many more results in this area are available in [4, 8, 16]. Recent research papers in this area can be found in [17].

In this study, scheduling problem of hybrid divisible load, i.e., a combination of arbitrarily divisible load and modularly divisible load, is considered. As is shown before, arbitrarily divisible load can almost always be scheduled optimally. However, modular load can never be optimally scheduled since it is an NP-complete problem. This paper derives the condition of optimal scheduling in a single-level tree network. Needless to say, the theory can be extensible to any topology such as bus, daisy-chain, hypercube networks, and so on. Once the problem formulation meets the condition the optimal fraction of load can be obtained optimally. The closed-form solution is provided in this paper. This work is just a starting point of optimal scheduling of hybrid divisible load theory. Thus, this paper just tries to convey basic concept of the hybrid divisible load theory.

2. PREVIOUS WORKS

The traditional scheduling of divisible loads depends on the architecture of the distributed computing system. For an example, we consider a single-level tree network with $(m+1)$ processors and m links, where the child processors $p_1, p_2, ..., p_m$ are connected to the root processor p_0 via links $l_1, l_2, ..., l_m$. The processing load originates at the root processor p_0. The root processor divides the total processing load into $m+1$ parts keeping the fraction α_0 for itself to process and/or compute and distributes the remaining load fractions $\alpha_1, \alpha_2, ..., \alpha_m$ to the child processors $p_1, p_2, ..., p_m$ one after other for processing. It means that the root processor distributes the load fractions to the other processors in the network in the sequence $p_1, p_2, ..., p_m$. Each processor in the network starts computing immediately upon receiving its load fraction. We now define the standard notations used in divisible load scheduling literature.

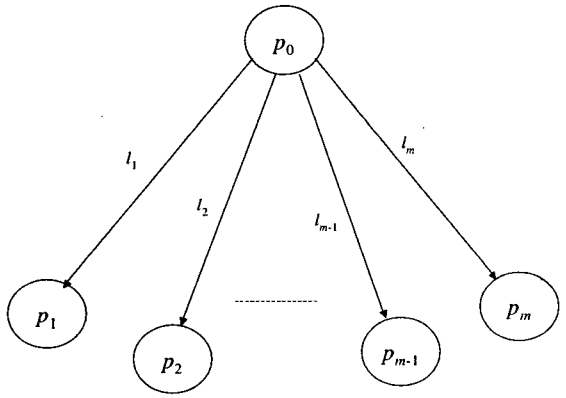

Figure 2. A sample of tree network processors

Notations

1) α_i: load fraction assigned to processor p_i;
2) w_i: ratio of the time taken by processor p_i, to compute a given load to the time taken by a standard processor, to compute the same load;
3) T_{cp}: time taken by a standard processor to process a unit load;
4) z_i: ratio of the time taken by communication link l_i, to communicate a given load to the time taken by a standard link to communicate the load;
5) T_{cm}: time taken by a standard communication link to send a unit load.

Based on these notations, we can see that $\alpha_i w_i T_{cp}$ is the time to process the load fraction α_i of the total processing load by the processor p_i. In the same way, $\alpha_i w_i T_{cm}$ is the time to communicate the load fraction α_i of the total processing load over the link l_i to the processor p_i. We can see that both $\alpha_i w_i T_{cp}$ and $\alpha_i w_i T_{cm}$ are in units of time. In divisible load scheduling literature, timing diagram (see Figure 3) is the usual way of representing the load distribution process. In this timing diagram the communication process is shown above the time axis and the computation process is shown below the time axis.

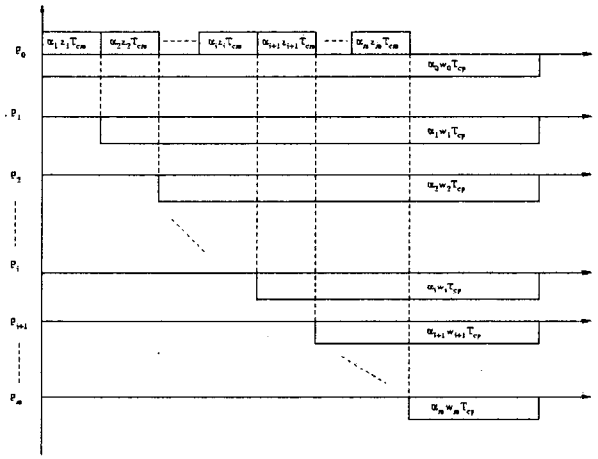

Figure 3. Timing diagram with front-end processors

The root processor may or may not be equipped with a front-end (a.k.a. communication co-processor). If the root processor is equipped with a front-end processor, communication of load fractions to other processors and the computation of its load fraction can be done at the same time. If the root processor is not equipped with a front-end, the root processor first distributes the load to other processors and then starts its computation.

Load fractions are derived based on the following set of equations:

$$\alpha_i w_i T_{cp} = \alpha_{i+1} w_{i+1} T_{cp} + \alpha_{i+1} z_{i+1} T_{cm} \quad i = 0,..,m-1 \quad (1)$$

and the normalization equation is given as follows:

$$\sum_{i=0}^{m} \alpha_i = 1. \quad (2)$$

The above equations can be rewritten as follows:

$$\alpha_i = \alpha_{i+1} f_{i+1} \quad i = 0,1,2,..,m-1, \qquad (3)$$

where

$$f_{i+1} = \frac{w_{i+1} T_{cp} + z_{i+1} T_{cm}}{w_i T_{cp}}. \qquad (4)$$

These recursive equations can be solved by expressing all the α_i (i=0,1,...,m-1) in terms of α_m as

$$\alpha_i = \alpha_m \prod_{j=i+1}^{m} f_j . \qquad (5)$$

From the normalization equation the fraction α_m can be expressed as

$$\alpha_m = \frac{1}{1 + \sum_{i=1}^{m} \prod_{k=i}^{m} f_k} \qquad (6)$$

and the load fraction α_i can be expressed as

$$\alpha_i = \frac{\prod_{j=i+1}^{m} f_j}{1 + \sum_{j=1}^{m} \prod_{k=j}^{m} f_k}. \qquad (7)$$

3. HYBRID SCHEDULING

Here, assume that two processors are available in a network. In addition, a modularly divisible load and an arbitrarily divisible load are placed in the root processor. The root processor should allocate appropriate fractions of load. The modularly divisible load means a load that cannot be divisible arbitrarily. For the simplicity of the mathematical modeling, assume that the modularity of the modularly divisible load is one, which means that it is actually an indivisible load. Root processor keeps the modularly divisible load and, if possible, some fractions of arbitrarily divisible load. It allocates the rest of the appropriate fraction of arbitrarily divisible load to the child processor. The problem is to decide the fractions of load including modularly divisible load.

Two-processor example conceptually simplifies the modeling of divisible load scheduling and is insightful. The two-processor example can recursively be extended to the general problems.

Assume that the length of the modularly divisible load is L, and that of arbitrarily divisible load is M. The length should be modeled or defined exactly. It can be an amount of data to be processed physically. However, its definition is beyond the scope of this paper. Figure 4 shows the length of inhomogeneous loads. The problem is to put the loads into fractions. Therefore, the physical length of load itself is meaningless. The two loads are put into one logical load (see Figure 4) and their lengths are normalized such that

$$b = \frac{L}{L+M}. \qquad (8)$$

Due to the property of inhomogeneous combination of loads, modular divisible one and arbitrarily divisible one, Equation (1) should be expressed more elaborately. By the traditional setting of divisible load, Equation (1) assumes that $O_{cp}(\alpha_i) = \alpha_i$ and $O_{cm}(\alpha_i) = \alpha_i$ where $O_{cp}(\alpha_i)$ is computational complexity of fraction α_i and $O_{cm}(\alpha_i)$ is communication complexity of fraction α_i. Then, two complexity terms can be incorporated into the following form:

$$O_{cp}(\alpha_i) w_i T_{cp} = O_{cp}(\alpha_{i+1}) w_{i+1} T_{cp} + O_{cm}(\alpha_{i+1}) z_{i+1} T_{cm} \quad i = 0,..,m-1 \qquad (9)$$

Under these assumptions that both complexities are linear such that $O_{cp}(\alpha_i) = \alpha_i$ and $O_{cm}(\alpha_i) = \alpha_i$ Equation (9) is exactly equal to Equation (1).

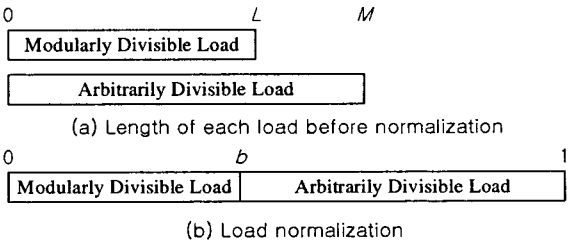

Figure 4. A hybrid load with normalized length 1.

Then the problem can be formulated in four cases regardless of the processing sequences. The two cases based on complexities are considered in the following subsections.

3.1. Homogeneous Case

First assume that the computing complexity and the communication complexity of both loads are the same and linear such that $O_{cp}(\alpha_i) = \alpha_i$ and $O_{cm}(\alpha_i) = \alpha_i$. Then, it is easy to calculate the fractions based on (1) and (2) or Equations (9) and (2). Then, we can straightforwardly get result as follows:

$$\alpha_0 = \frac{w_1 T_{cp} + z_1 T_{cm}}{w_0 T_{cp} + w_1 T_{cp} + z_1 T_{cm}} \qquad (10)$$

It is clear that if

$$\alpha_0 \geq b \qquad (11)$$

then the solution (10) is optimal. This fact is obviously clear.

3.2. Heterogeneous Case

Assume that the communication complexity of two loads is either same or different, and the computational complexity is either same or different. Then, Equations (9) and (2) can be rewritten to cope with two different loads as follows:

$$O_{cp}(b)w_0T_{cp} + O_{cp}(\alpha_0 - b)w_0T_{cp} = O_{cp}(1-\alpha_0)w_1T_{cp} + O_{cm}(1-\alpha_0)z_1T_{cm} \quad (12)$$

under the assumption of (11).

Solving Equation (12) is tricky. Thus, for the sake of simplicity, an example is provided.

Example: Assume that the computational complexity of modularly divisible one is given as $O_{cp}(b) = b$ and that of arbitrarily divisible part as $O_{cp}(\alpha_0 - b) = 2(\alpha_0 - b)$ and $O_{cp}(1-\alpha_0) = 2(1-\alpha_0)$, while $O_{cm}(1-\alpha_0) = 1-\alpha_0$. In this case, the optimal fraction is given as follows:

$$\alpha_0 = \frac{2w_1T_{cp} + z_1T_{cm} + bw_0T_{cp}}{2w_0T_{cp} + 2w_1T_{cp} + z_1T_{cm}}. \quad (13)$$

4. CONCLUSIONS

A new concept of hybrid divisible load theory is introduced in this paper. Hybrid system deals with a combination of modularly divisible load and arbitrarily divisible load. Main idea of hybrid divisible load theory is introduced with a simple example. A condition of optimality is derived for the hybrid case. An example of heterogeneous case is presented. Heterogeneous case is a generalized case where computational complexity and communication complexities are different. In this paper very simple case in considered. Thus, extension of this work will be done.

Acknowledgment

This work was in part supported financially by the MSRC-ITRC, Korea.

References

[1] R. Agrawal, and H.V. Jagadish, "Partitioning techniques for large-grained parallelism," *IEEE Transactions on Computers*, vol. 37, pp. 1627-1634, Dec. 1988.

[2] S. Bataineh, and T.G. Robertazzi, "Bus oriented load sharing for a network of sensor driven processors," *IEEE Transactions on Systems Man Cybernetics*, vol. 21, pp. 1202-1205, Sep-Oct 1991.

[3] V. Bharadwaj, D. Ghose, and V. Mani, "Optimal sequencing and arrangement in distributed single-level tree networks with communication delays," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, pp. 968-976, Sep. 1994.

[4] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems*, Los Alamitos, Calif., IEEE CS Press, 1996.

[5] J. Blazewicz, and M. Drozdowski, "The performance limits of a two-dimensional network of load sharing processors," *Foundations of Computing and Decision Sciences*, vol. 21, pp. 3-15, 1996.

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

[7] Y.C. Cheng, and T.G. Robertazzi, "Distr ㅛ ibuted computation for a tree network with communication delay," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, pp. 511-516, May 1990.

[8] M. Drozdowski, *Selected Problems of Scheduling Tasks in Multiprocessor Computer Systems*, Wydawnictwa Politechniki Poznanskiej, Poznan, Poland, 1997.

[9] D. Ghose, and H.J. Kim, "Load partitioning and trade-off study for large matrix-vector computations in multicast bus networks with communication delays," *Journal of Parallel and Distributed Computing*, vol. 55, pp. 32-59, Nov. 1998.

[10] D. Ghose, and H.J. Kim, Computing BLAS Level-2 operations on workstation clusters using the divisible load paradigm, *Mathematical and Computer Modelling*, *(Accepted; to appear)*

[11] D. Ghose and V. Mani, "Distributed computation with communication delays: Asymptotic performance analysis," *Journal of Parallel and Distributed Computing*, vol. 23, pp. 293-305, Dec. 1994.

[12] H.J. Kim, G.I. Lee, and J.G. Lee, "Optimal load distribution for tree network of processors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, pp. 607-612, April 1996.

[13] K. Li, "Parallel processing of divisible loads on partitionable static interconnection networks," *Cluster Computing*, Special Issue on: Divisible Load Scheduling, vol. 6, pp. 47-55, Jan. 2003.

[14] T.G. Robertazzi, "Processor equivalence for daisy chain load sharing processors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, pp.1216-1221, Oct. 1993.

[15] J. Sohn, and T.G. Robertazzi, "Optimal divisible job load sharing on bus networks," *IEEE Aerospace and Electronic Systems*, vol. 32, pp.34-40, Jan. 1996.

[16] Special Issue on: Divisible Load Scheduling, *Cluster Computing*, vol. 6, Jan. 2003.

[17] http://www.ee.sunysb.edu/tom/dlt.html.