Divisible Load Scheduling with Multiple Sources: Closed Form Solutions

Mequanint A. Moges,
Thomas G. Robertazzi[1]
Department of Electrical and
Computer Engineering
Stony Brook University
Stony Brook, NY 11794
e-mail:
(mmoges,tom)@ece.sunysb.edu

Dantong Yu
Department of Physics
Brookhaven National Laboratory
Upton, NY 11973
e-mail: dtyu@bnl.gov

Abstract —

Closed form solutions for optimal finish time and job allocation are largely obtained only for network topologies with a single load originating (root) processor. However, it often happens that load can be generated from multiple sources as in large-scale data intensive problems with geographically distributed resources. This paper introduces load scheduling strategy for tree networks with two load originating processors. A unique scheduling strategy that allows one to obtain closed form solutions for the optimal finish time and load allocation for each processor in the network is proposed.

I. Introduction

Today the scientific computation problems requiring intense problem-solving capabilities for problems arising from complex research and industrial application has driven in all global institution and industry segments the need for dynamic collaboration of many ubiquitous computing resources to be able to work together The problem of minimizing the processing time of extensive processing loads originated from various sources presents a great challenge that, if successfully met, could foster a range of new creative applications. Inspired by this challenge, we sought to apply divisible load theory to the problem of grid computing involving multiple sources connected to multiple sinks. So far research in this area includes [1] where tasks arrive according to a basic stochastic process to multiple nodes and [2] presents a first step technique for scheduling divisible loads from multiple sources to multiple sinks, with and without buffer capacity constraints.

Divisible load theory [3,4,5] is characterized by the fine granularity and large size of loads. There are also no precedence relations among the data elements. Such a load may be arbitrarily partitioned and distributed among processors and links in a system. The approach is particularly suited to the processing of very large data files in signal processing, image processing, experimental data processing, grid computing and computer utility applications.

There has been an increasing amount of study in divisible load theory since the original work of Cheng and Robertazzi [6] in 1988. The majority of these studies develop an efficient load distribution strategy and protocol in order to achieve optimal processing time in networks with a single root processor. The optimal solution is obtained by forcing the processors over a network to all stop processing simultaneously. Intuitively, this is because the solution could be improved by transferring load if some processors were idle while other are still busy [7]. Such studies for network topologies including linear daisy chains, tree and bus networks using a set of recursive equations were presented in [6,8,9] respectively. There have been further studies in terms of of load distribution policies for hypercubes [10] and mesh networks [11]. The concept of equivalent networks [12] was presented for complex networks such as multilevel tree networks. Work has also considered scheduling policy with multi-installment [13], multi-round algorithms [14], independent task scheduling [15], fixed communication charges [16], detailed parameterization and solution reporting time optimization [17] and combinatorial optimization [18]. Recently, though divisible load theory is fundamentally a deterministic theory, a study has been done to show some equivalence to Markov chain models [19].

Most of the earlier studies in the literature so far have assumed that the processing load originates form a single *root* processor. In a practical scenario, this need not be always true. In this paper, we relax this assumption and consider the case in which the processing load originates from two *root* processors. Our recent study [20] in this area considered two *root* processor model where optimal processing time solutions were obtained using a *linear programming* approach. This paper, unlike the previous research papers, presents closed form solutions using divisible load theory to tree networks with two load originating (root) processors. Applications include computational grids, a network of large number of loads and load sources with large number of transmission, processing and storage resources as shown in in Fig. 1. Computational grids aim at exploiting synergies that result from cooperation by sharing and aggregating distributed computational capabilities and delivering them as service.

Another example of grid problems is the case of high energy and nuclear physics experiments. Here large amounts of data originate from distributed researchers who must work closely together. The analysis of data in this type of experiments requires an increased computational power and network based computing platforms such as *Globus* [21] and *Condor* [22]. In such recently emerging platforms, scheduling and performance evaluation analysis for efficient use of distributed resources are important but challenging tasks. Effective scheduling of jobs in such a system is complicated by a number of factors including *data locality*, *network topology* and *latency*.

The organization of this paper is as follows. In section *II*, the system model used in this paper is discussed. The analysis of closed form solutions for an optimal finish time in single level tree networks with two load originating processors

[1]This work was supported by NSF Grant CCR-99-12331.

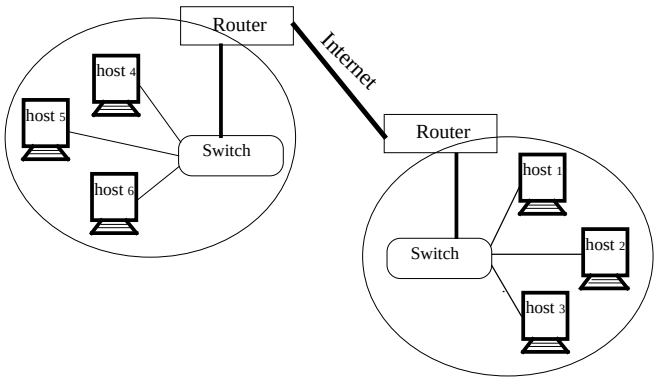

Figure 1: Simple computational grid made of two clusters.

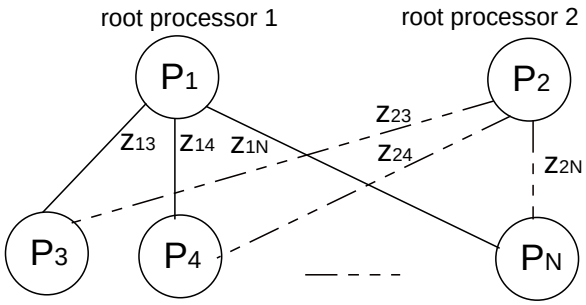

Figure 2: Single level tree network with two root processors.

is presented in sections III and IV. Section V presents the respective performance analysis results in terms of finish time and load assignment. Finally the conclusion appears in section VI.

II. Two Root Processors System Model

In this section, the various network parameters used in this paper are presented along with some notation and definitions. The network topology discussed in this study is a tree network consisting of two root processors (P_1 and P_2) and $N-2$ child processors (P_3, ... , P_N) with $2(N-2)$ links as shown in Fig. 2. It will be assumed that the total processing load considered here is of the arbitrarily divisible kind that can be partitioned into fractions of loads to be assigned to each processor over a network. The two root processors keep their own fraction of loads (α_1 and α_2) and communicate/distribute the other fractions of loads (α_3, α_4, ... α_N) assigned to the rest of processors in the network. Each processor begins to process its share of the load once the load share from either root processor has been completely received.

The load distribution strategy from either root processors to the child processors may be sequential or concurrent. In the sequential load distribution strategy, each root processor is able to communicate with only one child at a time. However, in the case of concurrent communication strategy, each root processor can communicate simultaneously/concurrently with all the child processors. The latter communication strategy can be implemented by using a processor which has a CPU that loads an output buffer for each output link. In this case it can be assumed that the CPU distributes the load to all of its output buffers at a rapid enough rate so that the buffer outputs are concurrent.

A Notations and Definitions:

L_i : Total processing load originated from root processor i, (i = 1, 2).

α_i : The total fraction of load that is assigned by the root processors to child i.

α_{1i} : The fraction of load that is assigned to processor i by the the first root processor.

α_{2i} : The fraction of load that is assigned to processor i by the second root processor.

$$\alpha_i = \alpha_{1i} + \alpha_{2i},\ i = 3, 4, ..., \text{N}.$$

ω_i : A constant that is inversely proportional to the processing speed of processor i in the network.

z_{1i} : A constant that is inversely proportional to the speed of the link between the first root processor and the i^{th} child processor in the network.

z_{2i} : A constant that is inversely proportional to the speed of the link between the second root processor and the i^{th} child processor in the network.

T_{cp} : Processing intensity constant. This is the time it takes the i^{th} processor to process the entire load when $\omega_i = 1$. The entire load can be processed on the i^{th} processor in time $\omega_i T_{cp}$.

T_{cm} : Communication intensity constant. This is the time it takes to transmit all the processing load over a link when $z_i = 1$. The entire load can be transmitted over the i^{th} link in time $z_i T_{cm}$.

T_i : The total time that elapses between the beginning of the scheduling process at $t = 0$ and the time when processor i completes its processing, $i = 1, ..., N$. This includes communication time, processing time and idle time.

T_f : This is the time when the last processor finishes processing.

$$T_f = \max(T_1, T_2, \ldots, T_N).$$

One convention that is followed in this paper is that the total load originating at the two root processors is assumed to be normalized to be a unit load. That is,

$$L_1 + L_2 = 1.$$

III. Single Level Tree Network with Two Root Processors and Concurrent Communication

The load scheduling strategy presented here targets finding solutions for optimal finish time (make-span) and job allocation in single level tree networks with two root processors.

Most previous load scheduling strategies in divisible load models can be solved algebraically in order to find the optimal finish time and load allocation to processors and links. In this case optimality is defined in the context of the specific interconnection topology and load distribution schedule used. An optimal solution is usually obtained by forcing all processors to finish computing at the same time. Intuitively, if there exist idle processors in the network, load can be transferred from busy processors to those idle processors [7].

The network topology considered here is a tree network with two root processors and $N-2$ child processors. In this case, it is assumed that the total processing load originates from the two root processors (P_1 and P_2). The scheduling strategy involves the partitioning and distribution of the processing loads originated from P_1 and P_2 to all the processors. The load distribution process proceeds as follows: the total processing loads originated from P_1 and P_2 are assumed to be L_1 and L_2 respectively. Each root processor keeps some fraction of the respective processing load for itself to compute and distributes the remaining load simultaneously to the child processors. The timing diagram shown in Fig. 3, shows the load distribution process discussed above. The figure shows that at time $t = 0$, the processors are all idle. The child processors start computation only after completely receiving their assigned fraction of load from either of the two root processors.

Now the equations that govern the relations among various variables and parameters in the network can be written as follows:

$$T_1 = \alpha_1\omega_1 T_{cp} \quad (1)$$

$$T_2 = \alpha_2\omega_2 T_{cp} \quad (2)$$

$$T_3 = (\alpha_{13}+\alpha_{23})\omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} \quad (3)$$

$$T_N = (\alpha_{1N}+\alpha_{2N})\omega_N T_{cp} + \alpha_{1N} z_{1N} T_{cm}. \quad (4)$$

As it was mentioned earlier, since total measurement load originating at the two root processors is assumed to be normalized to a unit load, the fractions of the total processing load should sum to one as:

$$L_1 + L_2 = 1 \quad (5)$$

$$\alpha_1 + \alpha_2 + \alpha_3 + ... + \alpha_{N-1} + \alpha_N = 1 \quad (6)$$

Since

$$L_1 = \alpha_1 + \sum_{j=3}^{N} \alpha_{1,j} \quad (7)$$

$$L_2 = \alpha_2 + \sum_{j=3}^{N} \alpha_{2,j} \quad (8)$$

The normalization equation given above can also be written in terms of the fraction of loads as:

$$\alpha_1 + \alpha_2 + \sum_{j=3}^{N} \alpha_{1,j} + \sum_{j=3}^{N} \alpha_{2,j} = 1 \quad (9)$$

As it can be seen from the timing diagram shown in Fig. 3, all processors stop processing at the same time, thus we have:

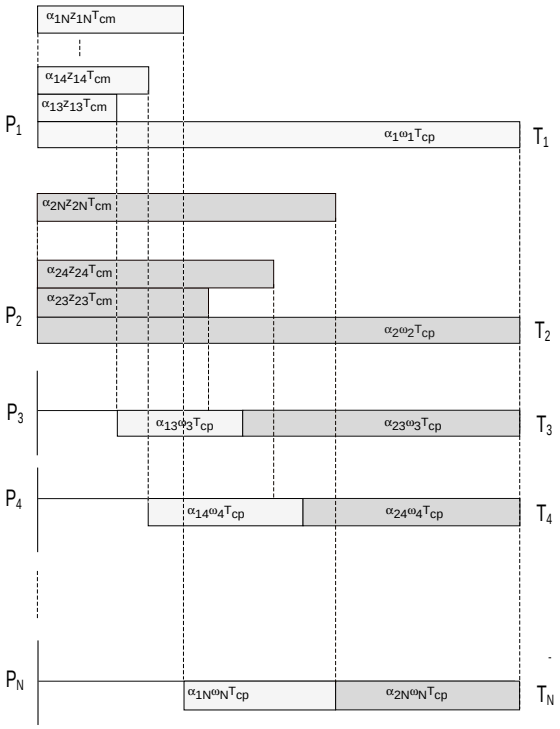

Figure 3: Timing diagram for a single level tree network with two root processors and concurrent communication.

$$T_1 = T_2 = T_3 = \ldots = T_N$$

Based on the above set of equations, one can write the following set of $N-1$ equations:

$$\alpha_1\omega_1 T_{cp} = \alpha_2\omega_2 T_{cp} \quad (10)$$

$$\alpha_2\omega_2 T_{cp} = \alpha_3\omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} \quad (11)$$

$$\alpha_3\omega_3 T_{cp} + \alpha_{13} z_{13} T_{cm} = \alpha_4\omega_4 T_{cp} + \alpha_{14} z_{14} T_{cm} \quad (12)$$

$$\alpha_{N-1}\omega_{N-1} T_{cp} + \alpha_{1N-1} z_{1N-1} T_{cm} = \alpha_N\omega_N T_{cp} + \alpha_{1N} z_{1N} T_{cm} \quad (13)$$

As it can be seen from the above set of equations, there is a smaller number of equations than the number of unknowns. Another $N-2$ equations can be written by setting up relationship between the fractions of loads within each child processor as:

$$\alpha_{23} z_{23} T_{cm} \leq \alpha_{13}(z_{13} T_{cm} + \omega_3 T_{cp}) \quad (14)$$

$$\alpha_{24} z_{24} T_{cm} \leq \alpha_{14}(z_{14} T_{cm} + \omega_4 T_{cp}) \quad (15)$$

$$\alpha_{2N} z_{2N} T_{cm} \leq \alpha_{1N}(z_{1N} T_{cm} + \omega_N T_{cp}) \quad (16)$$

In this case, there will be $2N-1$ equations (including the normalization equations) and $2N-2$ unknowns. The study in [20] has shown that such problem types can be solved by the use of *linear programming*.

IV. Closed Form Solutions

In order to obtain a closed form solution the following assumptions can be made regarding the load distribution strategy:

- The two root processors start to communicate with all of the child processors at the same time.

- For the same child P_1 terminates communication before P_2. It can be seen from this assumption that if the communication time of P_1 approaches to zero, then the network will be equivalent to a network with a single root processor.

- Each child processor starts processing after completely receiving its fraction of load received from either root processors.

- All processors are equipped with front-end processors, so that they will be able to communicate and process their respective load shares at the same time.

- The total communication and processing time of the fraction of load distributed by the first root processor (P_1) to each of the child processors is equal to the communication time needed to distribute the respective fractions of load by P_2 to each child processor. This can be achieved by controlling the transmission duration of P_2. Thus,

$$\alpha_{2i} z_{2i} T_{cm} = \alpha_{1i}(z_{1i} T_{cm} + \omega_i T_{cp}).$$

where $i > 2$.

The process of load distribution for this situation is shown in Fig. 4.

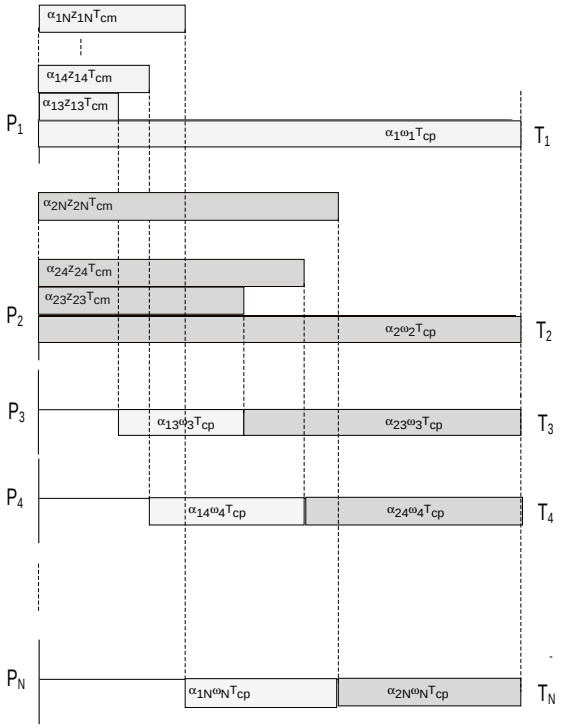

Figure 4: Timing diagram for a single level tree network with two root processors and concurrent communication : Scheduling for closed form solution.

Using the above set of equations and since for i > 2, $\alpha_i = \alpha_{1i} + \alpha_{2i}$, one can solve for α_{1i} and α_{2i} in terms of α_i as:

$$\alpha_{1i} = k_i \alpha_i \quad (17)$$

$$\alpha_{2i} = (1 - k_i)\alpha_i \quad (18)$$

where $k_i = z_{2i} T_{cm} / r_i$, and $r_i = z_{1i} T_{cm} + z_{2i} T_{cm} + \omega_i T_{cp}$.

All the above set of equations can be used to find the α_i's (i = 2, 3, ..., N) in terms of α_1 as:

$$\alpha_2 = (\omega_1 T_{cp} / \omega_2 T_{cp})\alpha_1 \quad (19)$$

$$\alpha_3 = s_3 \alpha_1 \quad (20)$$

$$\alpha_i = s_i \alpha_1 \quad (21)$$

where $s_i = (\omega_1 T_{cp} r_i)/(\omega_i T_{cp} r_i + z_{1i} T_{cm} z_{2i} T_{cm})$.

Now using the normalization equation, one can solve for α_1 as:

$$\alpha_1 = 1/(1 + (\omega_1 T_{cp}/\omega_2 T_{cp}) + \sum_{i=3}^{N} s_i) \quad (22)$$

The scheduler (P_1) will use the value of α_1 to obtain the amount of data that has to be processed by the rest of the $N - 1$ processors in the network.

The minimum processing time of the network will then be given as:

$$T_f = \omega_1 T_{cp}/(1 + (\omega_1 T_{cp}/\omega_2 T_{cp}) + \sum_{i=3}^{N} s_i) \quad (23)$$

For a homogeneous network with $\omega = 1$ and $T_{cp} = T_{cm} = 1$, the minimum processing time T_f approaches to $1/(Ns)$ as the number of processors N is made to be increasingly large. To see this result analytically, one can start from the expression: $\alpha_1 = 1/(1+(\omega_1/\omega_2)+\sum_{i=3}^{N} s_i)$. Now as N is made to be large, α_1 approaches $1/(2 + Ns)$ and since $N >> 2$, the expression for the minimum finish time will then be reduced to $1/(Ns)$.

V. Performance Analysis Results

This section presents the plots of load assignments to each processor vs. number of processors in a single level tree network with two root processors. In this case a homogeneous network is considered to study the effect of communication and computation speed variations and the number of processors on the load assignment.

The plot shown in Fig. 5, presents the load assignment to each of the processors in the network for the case when z_1 varies from 0.5 to 2.5 and z_2 is set to be fixed to 1.0.

The tree network that is used to obtain the plot in Fig. 5 has a homogeneous link and processor speed. In this case ω = 2 and the values of T_{cm} and T_{cp} are also set to be equal to one. The result shows that as the speed of the communication link becomes slower and slower the amount of load assigned to the child processors becomes less and less. In effect this will increase the total processing time of the system since the majority of the processing load is assigned to the root processor for computation.

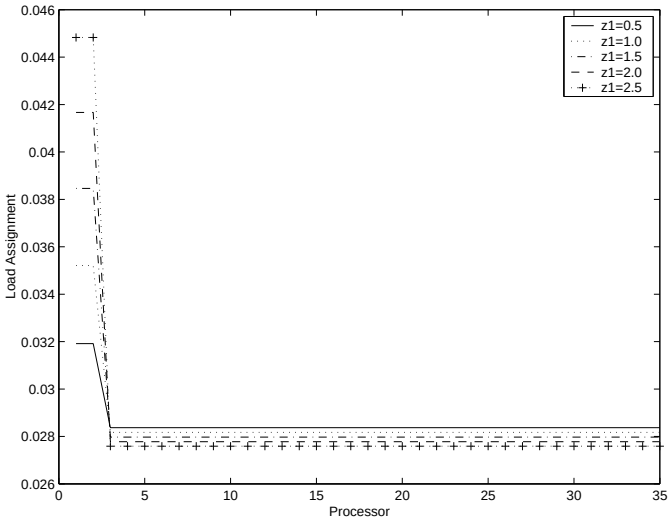

Figure 5: Load assignment when z_1 is varied and z_2 is fixed.

On the other hand Fig. 6 shows the same plot but for the case when z_1 is fixed and z_2 is varied from 0.5 to 2.5. For these parameters, the variation of the communication link between the second root processor and the child processors has no effect on the load assignment to each processor. This is because, as mentioned earlier, one of the constraints that is considered in this study is the case where the communication time of the second root processor and each of the child processors should be less or equal to the communication time between the first processor and each child processor plus the computation time of this fraction of load received from the first processor. That is, the job assigned from the second processor can only be processed after the first job assigned from the first processor is completely processed. If the second link is fast and the load is received earlier than the finish time of the first job, the new job has to wait until the previous job is done. This also enforces the fact that there will be no interruption during computation of the loads received from the two processors.

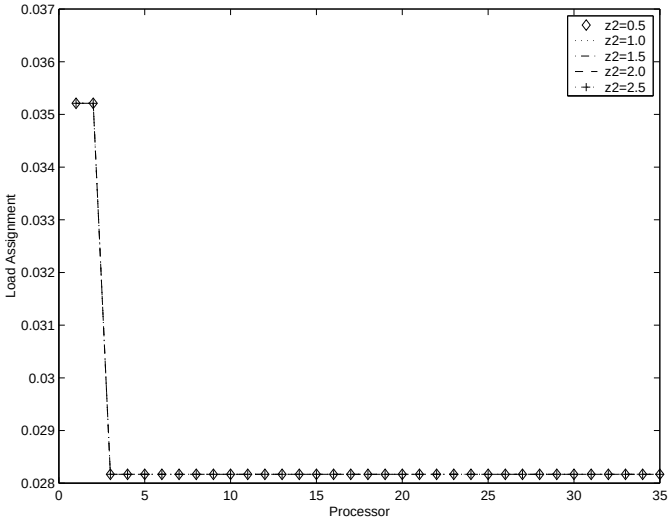

Figure 6: Load assignment when z_1 is fixed and z_2 is varied.

In Fig. 7, the finish time is plotted against the number of processors in the network for different inverse bus speeds, z_1 which is the communication link between the first root processor and the child processors. The communication link between the second root processor and the child processors is set to be fixed to $z_2 = 1$. The plot shows that as the number of processors is increased the minimum finish time approaches to $1/(Ns)$ which corresponds to the closed form solution analysis discussed earlier. It can also be seen from the same plot that the finish time can be minimized by increasing the speed of the communication link.

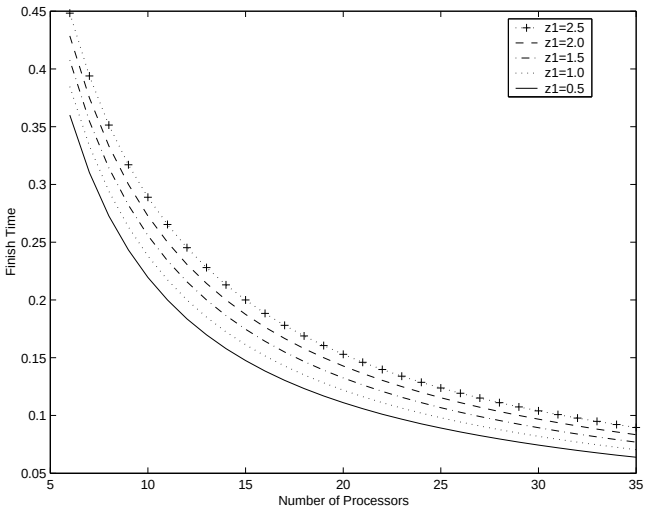

Figure 7: Finish time versus number of processors, for two root sources single level homogeneous tree network and variable inverse bus speed, z1, (first root processor links).

VI. Conclusion

In this paper, solutions for optimum allocation of loads to processors over a single level tree networks with two root processors are obtained. A new load scheduling strategy that results in a closed form solutions for an optimal allocation of loads to networks with two root processors is proposed. Using the closed form solutions obtained in this study one can easily study the performance analysis of tree networks in terms of the minimum finish time. Results showed that for the representative load scheduling strategy discussed in this paper, the minimum finish time approaches to $1/Ns$ as the number of processors in the network is made to be large enough.

Future work on this part may include other network topologies and load scheduling strategies used in divisible load theory. It will be also interesting to see the complexity involved in dealing with networks with more than two root processors.

References

[1] K. Ko and T.G. Robertazzi, "Scheduling in an environment of multiple job submissions," *Proceedings of the 2002 Conference on Information Sciences and Systems*, Princeton University, Princeton NJ, USA, 2002.

[2] H.M. Wong, B. Veeravalli, D. Yu and T.G. Robertazzi, "Data intensive grid scheduling: Multiple sources with capacity constraint," *IASTED International Conference on Parallel and*

Distributed Computing and Systems (PDCS 2003), Marina del Rey, CA, 2003.

[3] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, "Scheduling divisible loads in parallel and distributed systems," *IEEE Computer Society Press*, Los Alamitos, CA, 1996.

[4] V. Bharadwaj, D. Ghose, T.G. Robertazzi, "Divisible load theory: A new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, pp. 7–18, 2003.

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

[6] Y.C. Cheng and T.G. Robertazzi, "Distributed computation with communication delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 22, pp. 60–79, 1988.

[7] J. Sohn and T.G. Robertazzi, "Optimal divisible load sharing for bus networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32 pp. 34–40, 1996.

[8] Y.C. Cheng and T.G. Robertazzi, "Distributed computation for a tree network with communication delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, pp. 511–516, 1990.

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

[10] J. Blazewicz and M. Drozdowski, "Scheduling divisible jobs on hypercubes," *Parallel Computing*, vol. 21, pp. 1945–1956, 1996.

[11] 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.

[12] T.G. Robertazzi, "Processor equivalence for a linear daisy chain of load sharing processors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, pp. 1216–1221, 1993.

[13] V. Bharadwaj, D. Ghose, V. Mani, "Multi-installment load distribution in tree networks with delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, pp. 555–567, 1995.

[14] Y. Yang and H. Casanova, "UMR: A multi-round algorithm for scheduling divisible workloads," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[15] O. Beaumont, A. Legrand, and Y. Robert, "Optimal algorithms for scheduling divisible workloads on heterogeneous systems," *12^{th} Heterogeneous Computing Workshops*, HCW'2003, 2003.

[16] J. Blazewicz and M. Drozdowski, "Distributed processing of distributed jobs with communication startup costs," *Discrete Applied Mathematics*, vol. 76, pp. 21–41, 1997.

[17] A.L. Rosenberg, "Sharing partitionable workloads in heterogeneous NOWs: greedier is not better. In D.S. Katz, T. Sterling, M. Baker, L. Bergman, M. Paprzycki, and R. Buyya, editors," *Cluster Computing*, pp. 124–131, 2001.

[18] P.F. Dutot, "Divisible load on heterogeneous linear array," *Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[19] M. Moges and T.G. Robertazzi, "Optimal divisible load scheduling and Markov chain models," *Proceedings of the 2003 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, USA, 2003.

[20] M.A. Moges and T.G. Robertazzi, "Grid scheduling divisible loads from multiple sources via linear programming," *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, Cambridge, MA, 2004.

[21] I. Foster and C. Kesselman,"The Globus project: A status report," *In Proceedings of Heterogeneous Computing Workshop*, IEEE Press, pp. 4–18, 1998.

[22] M. Litzkow, M. Livny and M. Mutka, "A hunter of idle workstations," *In proceedings of the 8^{nd} International Conference on Distributed Computing Systems*, pp. 104–111, 1988.