Scheduling Second Order Computational Load in Master-Slave Paradigm

S. Suresh, *Senior Member, IEEE*, Cui Run, Hyoung Joong Kim, *Member, IEEE*,
Thomas G. Robertazzi, *Fellow, IEEE*, Young-Il Kim

Abstract

Scheduling divisible loads with the nonlinear computational complexity is a challenging task as the recursive equations are nonlinear and it is difficult to find closed-form expression for processing time and load fractions. In this study, we attempt to address a divisible load scheduling problem for computational loads having second order computational complexity in a master-slave paradigm with non-blocking mode of communication. First, we develop algebraic means of determining the optimal size of load fractions assigned to the processors in the network using a mild assumption on communication to computation speed ratio. We use numerical simulation to verify the closeness of the proposed solution. Like in earlier works which consider processing loads with first order computational complexity, we study the conditions for optimal sequence and arrangements using the closed-form expression for optimal processing time. Our finding revels that the condition for optimal sequence and arrangements for second order computational loads are the same as that of linear computational loads. This scheduling algorithm can be used for aerospace applications such as Hough transform for image processing and pattern recognition using hidden Markov model.

Index Terms

Polynomial computational loads, processing time, divisible load theory, master-slave paradigm, optimal sequence and arrangement, pattern recognition.

I. Introduction

RESEARCHERS are producing huge amount of data to solve complex and interdisciplinary problems. The efforts to solve such complex problems are hindered by time consuming post-processing in a single workstation. Data-driven computation is an active area of research, which addresses the issue of handling huge data set. The main objective in data-driven computation is to minimize the processing time of computing loads by using distributed computing system. These computing loads are assumed to be divisible arbitrarily into small fractions and processed independently in the processors. The above assumption on computing loads is suitable for many practical applications involving data parallelism such as image processing, pattern recognition, bio-informatics, data mining, etc. The main thrust in the parallel processing of divisible loads is to design efficient scheduling algorithms that minimize the total load processing time. The domain of scheduling divisible loads in multiprocessor system is commonly referred as Divisible Load Theory (DLT) and is of interest to researchers in the field of scheduling loads in computer networks. The problem of scheduling divisible loads in intelligent sensor network started in 1988 by Cheng and Robertazzi [13]. Here, an intelligent sensor network with master-slave architecture is considered where a master processor can measure, compute and communicate with other intelligent sensors for collaborative computing.

The first mathematical model considered [13] is similar to a linear network of processors. The optimal load allocation strategy presented in [13] is extended to tree networks in [14] and bus networks in [11], [31]. An optimal load allocation for linear network of processors is presented by the theory of all processors stop computing at the same time instant [13]. In fact, this condition has been shown to be a necessary and sufficient condition for obtaining optimal processing time in linear networks [30] by using the concept of processor equivalence. An analytical proof of this assumption in bus networks is presented [32]. This assumption has been proven in a rigorous manner and it is shown that this assumption is true only in a restricted sense [8]. The concepts of optimal sequencing and optimal arrangement are introduced [4], [26]. Parameters for computation and communication are probed for adaptive distributed processing [20].

Since, 1988 research works [6]–[18], [20], [22], [26], [30]–[34], [38] in divisible load theory framework have been carried out by algebraic means to determine optimal fractions of a load distributed to processors in the network such that the total load processing time is minimum. A number of scheduling policies have been investigated including multi-installments [5], multi-round scheduling [7], [39], multiple loads [15], limited memory [18], [35], simultaneous distribution [21], [29], simultaneous start [33], start-up delay [9], [36], detailed parameterizations and solution time optimization [1], and combinatorial schedule

Cui Run is a Ph.D. student of CIST, Graduate School of Information Management and Security, Korea University, Seoul 136-701, Korea, e-mail: (cuirun@korea.ac.kr).

S. Suresh is an Assistant Professor, School of Computer Engineering, Nanyang Technological University, Singapore, e-mail: (ssundaram@ntu.edu.sg).

Hyoung Joong Kim is a Professor of CIST, Graduate School of Information Management and Security, Korea University, Seoul 136-701, Korea, e-mail: (khj-@korea.ac.kr).

Thomas G. Robertazzi is a Professor of Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, New York 11794-2350, USA, e-mail: (tom@ece.sunysb.edu).

Young-Il Kim is a Vice President of Korea Telecom, KT Central R&D Center, Seoul 137-792, Korea, e-mail: (yikim@kt.com).

optimization [19]. Divisible loads may be divisible in fact or as an approximation as in the case of a large number of relatively small independent tasks [3], [10]. Recently, ten reasons to use the concept of divisible load scheduling theory are presented [31]. Results and open problems in divisible load scheduling in single level tree network is highlighted in [6]. A complete survey and results in divisible load scheduling algorithm can be found [8], [31], [33]. Aforementioned research works in the domain of divisible load scheduling in distributed systems consider processing load requiring linear computational power.

There is an increasing amount of research on real-time modeling and simulation of complex systems such as nuclear modeling, aircraft/spacecraft simulation, biological system, bio-physical modeling, genome search, etc. It is well known that many algorithms requires nonlinear computational complexity, i.e., the computational time of the given data/load is a nonlinear function of the load size (N). For the first time in the literature, a nonlinear cost function is considered [17], [22]. In [22], the computational loads require nonlinear processing time depending on the size of load fractions. It has been mentioned that because of nonlinear dependency the speed-up achieved by simultaneous-start is super-linear [17], [22]. Finding algebraic solution for nonlinear computational loads is a challenging issue. In this paper, we present approximate algebraic solution for second order computational loads.

Image processing and pattern analysis for aerospace applications of which computational complexity is $O(N^2)$ include line detection using the Hough transform [40], and pattern recognition using 2D hidden Markov model (HMM) [28]. The classical Hough transform was concerned with the identification of lines in the image, but later this transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The computational complexity for N points is approximately proportional to N^2. When, N is large, parallel or distributed processing is desired [42]. A separable 2D HMM for face recognition builds on an assumption of conditional independence in the relationship between adjacent blocks. This allows the state transition to be separated into vertical and horizontal state transitions. This separation of state transitions brings the complexity of the hidden layer of the proposed model from the order of $O(N^3k)$ to the order of $O(N^2k)$, where N is the number of the states in the model and k is the total number of observation blocks in the image [42]. In addition, we can also find real-world problems like molecular dynamic simulation of macromolecular systems, learning vector quantization neural network [24] and block tri-diagonalization of real symmetric matrices [2] which require second order computational complexity.

In this paper, we address the scheduling problem for second order computational loads in a master-slave paradigm with non-blocking mode communication. Here, the second order time complexity computational load arrives at master processor and master processor distributes the load fractions one-by-one to the slave processors in the network using non-blocking mode of communication. Using a mild assumption on the communication to computation speed ratio and the minimum granularity of any load fractions, we derive an algebraic solution for the optimal size of the each load fraction and the total load processing time. Numerical solutions are compared with the algebraic solution to see if they conform to each other. The results clearly indicate that the algebraic closed-form expression matches closely with the numerical solution. Finally, we study the conditions for optimal sequence and optimal arrangement using the closed-form expression. Our finding reveals that the condition for optimal sequence/arrangements is the same as that of linear computational loads.

II. Mathematical Formulation

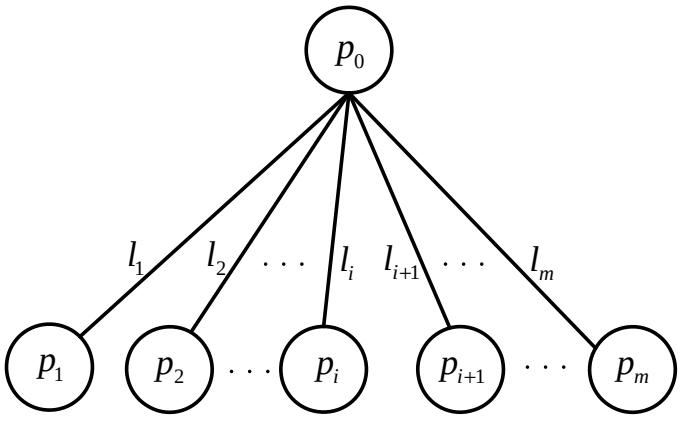

Fig. 1. Master-Slave Network

In this section, we describe the master-slave model and formulate the problem. We consider a second order computational load which is arbitrarily divisible. The user submits the computational load in the master processor (p_0). The master processor p_0 is connected to m slave processors ($p_1, p_2, \cdots, p_m$) through the links ($l_1, l_2, \cdots, l_m$) as shown in Fig. 1. The root processor

(p_0) divides the processing load is into $m+1$ fractions (α_0, α_1, $\cdots$, α_m), keeps α_0 for itself and distributes the remaining m fractions to child processors (p_0, p_1, p_2, $\cdots$, p_m) in the network. The processing time to compute the load fraction depends linearly on the computing speed of the processor and nonlinearly in terms of the size of load fraction. In this paper, we use the non-blocking mode of communication [25], [37] to distribute the load fractions (α_0, α_1, $\cdots$, α_m) to slave processors (p_1, p_2, $\cdots$, p_m). In non-blocking mode of communication, the child processor will start the computation process while its front-end starts receiving the fraction of loads. The objective of this study is to find the optimal size of load fractions assigned to the processors in the network such that the total processing time is minimum. The following are the notations used in this paper.

α_0: fraction of the load assigned to the root processor p_0.
α_i: fraction of the load assigned to the child processor p_i.
A_i: inverse computing speed on the processor p_i.
G_i: inverse link speed on the link l_i.
$T(m)$: total time taken to process the complete load.
N: total size of the load fractions.
m: number of the slave processors.
n: order of processing.
δ: minimum granularity of any load fraction.

A. Optimal Load Scheduling

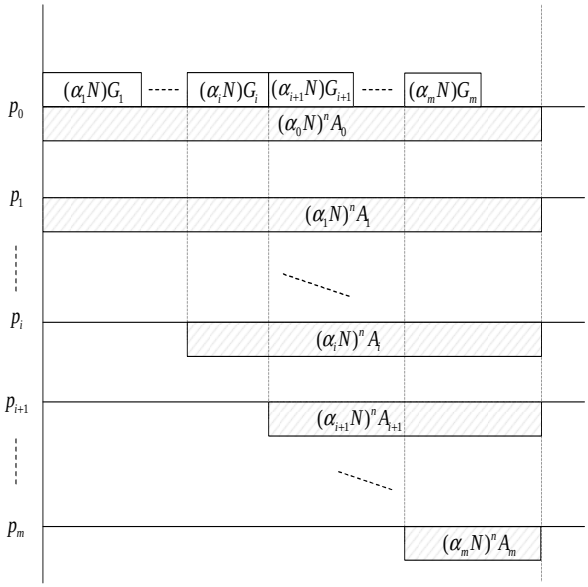

Fig. 2. Timing diagram describing the load distribution process in master-slave network

We will derive the closed-form expressions for the load fractions and processing time for nonlinear processing load in the non-blocking mode of communication model. For the purpose of derivation of the closed form expression, we consider a sequence of load distribution, p_1, p_2, $\cdots$, p_m, in that order. The problem is to find the optimal sizes of the load fractions that are assigned to the processors in the network such that the final processing time is minimal. The load distribution process by the master processor p_0 is illustrated by means of a timing diagram as shown in Fig. 2. As in the case of linear computational loads [8], the processing time for nonlinear computational loads is minimum only when all processors stop computing at the same time. The detailed proof for second order computational loads is given in the Appendix I.

From the timing diagram, we can write the recursive load distribution equations as follows:

$$(\alpha_1 N)^n A_1 = (\alpha_o N)^n A_0, \tag{1}$$

$$(\alpha_{i+1} N)^n A_{i+1} + (\alpha_i N) G_i = (\alpha_i N)^n A_i, \quad i = 1, 2, \cdots, m-1. \tag{2}$$

The above equations are reduced to

$$(\alpha_1 N)^n = (\alpha_o N)^n f_1, \tag{3}$$

$$(\alpha_{i+1} N)^n = (\alpha_i N)^n f_{i+1} - (\alpha_i N)\beta_i f_{i+1}, \quad i = 1, 2, \cdots, m-1, \tag{4}$$

where

$$f_{i+1} = \frac{A_i}{A_{i+1}} \quad i = 0, 1, 2, \cdots, m-1, \tag{5}$$

$$\beta_i = \frac{G_i}{A_i} \quad i = 1, 2, \cdots, m-1. \tag{6}$$

The normalization equation is

$$\sum_{i=0}^{m} \alpha_i = 1. \tag{7}$$

Equations (3) and (4) can be reduced to

$$\alpha_1 N = \alpha_0 N \sqrt[n]{f_1}, \tag{8}$$

$$\alpha_{i+1} N = \alpha_i N \sqrt[n]{f_{i+1}} \left[1 - \frac{\beta_i}{(\alpha_i N)^{n-1}}\right]^{1/n}, \quad i = 1, 2, \cdots, m-1. \tag{9}$$

The size of load fractions can be obtained by substituting Equations (8) and (9) in Equation (7) and solved analytically. Solving these equations is difficult and computationally intensive. In this paper, we derive a closed-form expression for the size of load fraction and processing time by approximating the terms inside the root. Finding approximate closed-form expression for higher power is difficult. Hence, in this paper, we consider only the second power ($n = 2$). If we substitute n with 2 in Equations (8) and (9), then the equations are reduced as

$$\alpha_1 N = \alpha_0 N \sqrt{f_1}, \tag{10}$$

$$\alpha_{i+1} N = \alpha_i N \sqrt{f_{i+1}} \sqrt{1 - \frac{\beta_i}{\alpha_i N}}, \quad i = 1, 2, \cdots, m-1. \tag{11}$$

Assumption: We assume that the ratio of communication time to computation (β_i) is very small in most practical distributed systems. Also, the size of load fraction assigned to the child processor $\alpha_i N$ is larger than β_i.

Using the above assumption, we will express the term ($\sqrt{1 - \frac{\beta_i}{\alpha_i N}}$) in Equation (11) in Taylor series as

$$\sqrt{1 - \frac{\beta_i}{\alpha_i N}} = 1 - \frac{\beta_i}{(\alpha_i N)} + O\left(\left(\frac{\beta_i}{\alpha_i N}\right)^2\right) \tag{12}$$

Note that the communication-to-computation ratio (β_i) is less than 1 and the load fraction assigned to the child processor is greater than the minimum granularity of processing load ($\alpha_i N > \delta$). Hence, the higher order terms of $\frac{\beta_i}{\alpha_i N}$ are small and are neglected.

In this paper, we consider a first-order approximation of square root to derive the closed-form expression.

$$\sqrt{1 - \frac{\beta_i}{\alpha_i N}} \approx 1 - \frac{\beta_i}{2\alpha_i N} \tag{13}$$

The approximation holds only when $\frac{\beta_i}{\alpha_i N}$ is much smaller than one and $\frac{\beta_i}{\alpha_i N}$ moves closer to $\frac{\beta_i}{\delta}$, the approximation become worse.

By substituting the approximation of the square root, Equation (11) can be simplified as

$$\alpha_{i+1} N \approx \alpha_i N \sqrt{f_{i+1}} - \frac{\beta_i \sqrt{f_{i+1}}}{2}, \quad i = 1, 2, \cdots, m-1. \tag{14}$$

By substituting Equations (14), and (10) in normalization Equation (7), we can derive the closed-form expression for the load fraction α_0 assigned to the root processor p_0 as

$$\alpha_0 = \frac{N + x(m)}{N y(m)}, \tag{15}$$

where

$$x(m) = \frac{1}{2} \sum_{i=1}^{m-1} \beta_i \left[\sum_{j=i+1}^{m} \prod_{k=i+1}^{j} \sqrt{f_k}\right], \tag{16}$$

$$y(m) = 1 + \left[\sum_{i=1}^{m} \prod_{j=1}^{i} \sqrt{f_j}\right]. \tag{17}$$

From Equations (10) and (15), the load fraction α_i can be expressed in terms of load fraction α_0 as

$$\alpha_i N \approx \alpha_0 N \sqrt{f_1 f_2 \cdots f_i} - \frac{1}{2} \sum_{j=1}^{i-1} \beta_j \prod_{k=j+1}^{i} \sqrt{f_k}, \quad i = 1, 2, \cdots, m. \tag{18}$$

By substituting the closed-form expression for load fraction α_0 in Equation (18), one can easily calculate the size of load fraction assigned to any processor in the network as follows:

$$\alpha_i = \frac{1}{N} \left[\frac{N + x(m)}{y(m)} \sqrt{f_1 f_2 \cdots f_i} - \frac{1}{2} \sum_{j=1}^{i-1} \beta_j \prod_{k=j+1}^{i} \sqrt{f_k} \right], \quad i = 1, 2, \cdots, m. \tag{19}$$

Now, we derive the closed-form expression for the total load processing time. From the timing diagram shown in Fig. 2, the total load processing time $T(m)$ is given as follows:

$$T(m) = (\alpha_0 N)^2 A_0 = \left[\frac{N + x(m)}{y(m)} \right]^2 A_0. \tag{20}$$

One should remember that the above closed-form expression for processing time is derived under the assumption that the communication time is less than the computation time. When the communication time is greater than the computation time ($\beta_i > 1$), simultaneous processing is not possible. The processor will have cycles of the work and wait period. For this case, finding closed-form expression is not straightforward. This case can be handled easily using the equivalent processor concept explained in [25], [37].

The advantage of the closed-form expression is that we can directly derive conditions for the optimal sequence of load distribution and the optimal arrangement of processors. Before analyzing the theoretical results, we present a numerical example to understand the characteristics of nonlinear divisible load theory with non-blocking mode of communication.

B. Numerical Example 1.

Consider the task of finding ellipses in an 512×512 image. Lets assume that the ellipses are oriented along the principle axes. Hence, we need four parameters ($k = 4$) (two for the center of the ellipse and two for the radii) to describe the ellipse. The computational complexity in identifying the ellipse is $O(N^{k-2})$, which is $O(N^2)$. Here, N is image space ($N = 262144$). For simplicity, we consider a small region of interest 10×10 ($N = 100$) in our example. The root processor divides the image size into small fractions and distribute them to child processors. Each child processor computes the Hough space for a given resolution and generates the accumulator array for their fraction of image region. The size of accumulator array depends on the resolution and does not depends on the image size. Finally, the root processor collects all the arrays and identify the candidate points for ellipses. For simplicity, we neglect the result collection time (resolution is much smaller than image size) from each processor.

Consider a single-level tree network with three processors ($m = 3$). The time to compute the accumulator array for one pixel (processors parameter) and the time to communicate one pixel through the link (link parameters) are given in Table I. The total size of load fraction N is assumed to be 100 units. Using the closed-form expression, the values of fractions assigned to the processors are computed as follows: $\alpha_0 = 0.12840$, $\alpha_1 = 0.13619$, $\alpha_2 = 0.35132$ and $\alpha_3 = 0.38480$. The corresponding total load processing time is 148,384 units of time. The total load processing time obtained by analytically solving the nonlinear recursive equations using nonlinear least square solver [41] is 148,170 units of time. The load fractions obtained using analytical solution are: $\alpha_0 = 0.128309$, $\alpha_1 = 0.13609$, $\alpha_2 = 0.351068$, and $\alpha_3 = 0.38453$. From the results, we can see that the closed-form expressions closely approximate the actual solution.

TABLE I
PROCESSOR AND COMMUNICATION LINK PARAMETERS USED IN THE NUMERICAL EXAMPLE 1.

Parameters	P_0	P_1	P_2	P_3
A	900	800	120	100
G	-	20	1	0.85

The processing time obtained using the closed-form expression and actual solution obtained using the analytical solution are given in Table II. From the table, we can see that the processing time obtained using the approximate closed-form solution is matching with the analytical solution. The difference between the solutions depends on the ratio between communication time to computation time (β_i) and size of load fraction ($\alpha_i N$). The error is small when $\frac{\beta_i}{\alpha_i N}$ close to zero and it becomes worse when $\frac{\beta_i}{\alpha_i N}$ moves closer to $\frac{\beta_i}{\delta}$.

The main objective of deriving the closed-form expression is to study the behavior of second order load scheduling problems. In the following section, we show that the approximate closed-form solution can be directly used to find the conditions for optimal arrangements and optimal sequence of load distribution.

TABLE II
TOTAL LOAD PROCESSING TIME OBTAINED USING ANALYTICAL SOLUTION OF RECURSIVE EQUATIONS AND APPROXIMATE CLOSED-FORM EXPRESSION.

# of child processors	Approximate solution	Analytical solution
1	2,119,482	2,119,482
2	391,247	390,995
3	148,384	148,170

C. Homogeneous System

As a special case for homogeneous system ($A_i = A$ and $G_i = G$), the load fraction assigned to the root processor (α_0) is obtained by substituting $f_i = 1$ and $\beta_i = \beta$ in Equation (15) as follows:

$$\alpha_0 = \frac{4N + m\beta(m-1)}{4N(m+1)}. \tag{21}$$

Load fraction assigned to any child processor p_i is obtained as follows:

$$\alpha_i = \frac{4N + m\beta(m-1)}{4N(m+1)} - \frac{(i-1)\beta}{2N} \qquad i = 1, 2, \cdots, m. \tag{22}$$

The total load processing time for homogeneous system is computed as follows:

$$T(m) = \left[\frac{4N + m\beta(m-1)}{4(m+1)}\right]^2 A. \tag{23}$$

In the homogeneous case, if the communication to computation ratio tends to be zero, the load fractions assigned to the processors converge to equal load fraction, i.e.,

$$\alpha_0 = \lim_{\beta \to 0} \frac{4N + m\beta(m-1)}{4N(m+1)} = \frac{1}{m+1} \tag{24}$$

and

$$\alpha_i = \lim_{\beta \to 0} \left[\frac{4N + m\beta(m-1)}{4N(m+1)} - \frac{(i-1)i\beta}{2N}\right] = \frac{1}{m+1} \qquad i = 1, 2, \cdots, m, \tag{25}$$

and the total load processing time converges to

$$T(m) = \left[\frac{N}{m+1}\right]^2 A. \tag{26}$$

From Equation (26), we can see that the total processing time is super-linear with increase in the number of processors.

III. OPTIMAL SEQUENCE OF LOAD DISTRIBUTION

In the linear divisible load theory, the closed-form expression is used to find the condition for the optimal sequence of load distribution. Similarly, one needs to derive the closed-form expression to study the behavior of the nonlinear divisible load condition. In this section, we present the condition for optimal sequence of load distribution obtained from the approximate closed-form expression. First, we present an example to understand the effect of changing the sequence of load distribution and later generalize the result. For this purpose, we consider a three-processor ($m = 3$) network. From Equation (20), we can see that the processing time is a function of load fraction α_0 assigned to the processor p_0. Hence, it is sufficient to analyze the behavior of α_0 instead of processing time $T(m)$.

Case A: The sequence of load distribution is (p_1, p_2, p_3), i.e., the root processor p_0 first sends the load fraction to the processor p_1, next to the processor p_2, and last to the processor p_3. Using the closed-form expression, we can write α_0 as

$$\alpha_0 N = \frac{N + \beta_1(\sqrt{f_2} + \sqrt{f_2 f_3})/2 + \beta_2(\sqrt{f_3})/2}{1 + \sqrt{f_1} + \sqrt{f_1 f_2} + \sqrt{f_1 f_2 f_3}}. \tag{27}$$

The above equation can be expressed in terms of system parameters (A_i, G_i) as

$$\alpha_0 N = \frac{2N\sqrt{A_1A_2A_3} + G_1(\sqrt{A_2} + \sqrt{A_3}) + G_2\sqrt{A_1}}{2(\sqrt{A_1A_2A_3} + \sqrt{A_0A_2A_3} + \sqrt{A_0A_1A_3} + \sqrt{A_0A_1A_2})}. \tag{28}$$

Case B: Now, we change the load distribution sequence as (p_1, p_3, p_2), i.e., the root processor p_0 first sends the load fraction to the processor p_1, next, to the processor p_3 and finally to the processor p_2. The load fraction (α_0') can be obtained by interchanging (A_2, G_2) and (A_3, G_3) in earlier expression.

$$\alpha_0' N = \frac{2N\sqrt{A_1A_2A_3} + G_1(\sqrt{A_2} + \sqrt{A_3}) + G_3\sqrt{A_1}}{2(\sqrt{A_1A_2A_3} + \sqrt{A_0A_2A_3} + \sqrt{A_0A_1A_3} + \sqrt{A_0A_1A_2})}. \tag{29}$$

Now, we have to find the condition for $\alpha_0 \leq \alpha_0'$. By subtracting Equation (29) and Equation (28), we get

$$\alpha_0 N - \alpha_0' N = \frac{\sqrt{A_1}(G_2 - G_3)}{2(\sqrt{A_1A_2A_3} + \sqrt{A_0A_2A_3} + \sqrt{A_0A_1A_3} + \sqrt{A_0A_1A_2})}. \tag{30}$$

From the above equation, we can say that the total load processing time is minimal for load distribution sequence (p_1, p_2, p_3) if and only if G_2 is less than G_3. From the results obtained for the three-processor network case, we can generalize the result as follows:

Optimal Sequencing Theorem: Given an $(m+1)$-processor single-level tree network with non-blocking mode of communication, the optimal sequence of load distribution is produced if the root processor distributes the load fractions in ascending order of communication speed parameter G_i of the links.

Proof: For m processors, consider a case when the root processor p_0 distributes the load fractions to child processors in the following sequence (p_1, p_2, $\cdots$, p_{i-1}, p_i, p_{i+1}, $\cdots$, p_m). The value of load fraction α_0 assigned to root processor for this sequence is

$$\alpha_0 = \frac{N + x(m)}{Ny(m)}. \tag{31}$$

Consider another sequence of load distribution where the root processor distributes the load fractions to child processors in a sequence (p_1, p_2, $\cdots$, p_{i-1}, p_{i+1}, p_i, $\cdots$, p_m). The value of load fractions assigned to the root processor in this sequence is

$$\alpha_0' = \frac{N + x'(m)}{Ny'(m)}. \tag{32}$$

The load fraction for the new sequence can be obtained by exchanging the (G_i, A_i) and (G_{i+1}, A_{i+1}) in Equation (31). The interchange affects terms f_i, f_{i+1}, f_{i+2}, β_i and β_{i+1} only, and does not affect the other terms. Note that, because of this interchange, $y(m)$ and $y'(m)$ will not change. Now, we will find the conditions for $\alpha_0 \leq \alpha_0'$, which is the same as $x(m) \leq x'(m)$. The terms $x(m)$ and $x'(m)$ are a function of f and β.

$$x(m) = \frac{1}{2}\left\{ \begin{array}{c} \beta_1\left[\sqrt{f_2} + \sqrt{f_2f_3} + \cdots + \sqrt{f_2f_3\cdots f_m}\right] + \cdots \\ +\beta_i\left[\sqrt{f_{i+1}} + \sqrt{f_{i+1}f_{i+2}} + \cdots + \sqrt{f_{i+1}f_{i+2}\cdots f_m}\right] \\ + \cdots + \beta_{m-1}\sqrt{f_m} \end{array} \right. \tag{33}$$

Now, $x(m) - x'(m)$ is given as follows:

$$x(m) - x'(m) = \frac{G_i - G_{i+1}}{2\sqrt{A_iA_{i+1}}}. \tag{34}$$

Then,

$$\alpha_0 N - \alpha_0' N = \frac{G_i - G_{i+1}}{2y(m)\sqrt{A_iA_{i+1}}}. \tag{35}$$

Here, note that $\alpha_0 N \leq \alpha_0' N$ only when $G_i \leq G_{i+1}$. By recursively applying the above condition, we can get the optimal load distribution sequence which satisfies the condition $G_1 \leq G_2 \leq \cdots \leq G_m$. This proves the theorem.

The result obtained from the *Optimal Sequencing Theorem* is similar to that of the optimal sequence of load distribution presented for the linear case [8], [26].

Numerical Example 2.: In this example, we consider the same parameters used in the Numerical Example 1. In the previous example, we have used load distribution sequence (p_1, p_2, p_3). The total load processing time is 148,384 units. By applying the optimal sequencing theorem, the optimal sequence of load distribution is (p_3, p_2, p_1). The load fractions assigned to the processors in the network are $\alpha_0 = 0.128236$, $\alpha_1 = 0.136015$, $\alpha_2 = 0.351175$, and $\alpha_3 = 0.38465$. The total load processing time is 148,000 units. From this result, we can see that the total processing time for the optimal sequence is less than that for the previous sequence.

IV. Optimal Arrangement of Processors

In this section, we derive the condition for the optimal arrangement of processors in the nonlinear divisible load problem using our closed-form expressions. First, we present an example to understand the effect of changing the processor arrangement and later generalize the result. For this purpose, we consider a three-processor ($m = 3$) network. Here, the sequence of load distribution is fixed as (p_1, p_2, p_3).

Case A: The processor p_1 is connected to link l_1, processor p_2 is connected to link l_2, and processor p_3 is connected to link l_3. Using our closed-form expression, we can write α_0 as Equation (28).

Case B: Now, we change the arrangement of processors in the network. The processor p_1 is connected to link l_2 and the processor p_2 is connected to link l_1. The load fraction (α_0') can be obtained by interchanging A_1 and A_2 in the earlier expression as Equation (28).

$$\alpha_0' N = \frac{2N\sqrt{A_1A_2A_3} + G_1(\sqrt{A_1} + \sqrt{A_3}) + G_2\sqrt{A_2}}{2(\sqrt{A_1A_2A_3} + \sqrt{A_0A_2A_3} + \sqrt{A_0A_1A_3} + \sqrt{A_0A_1A_2})}. \tag{36}$$

Now, we have to find the condition for $\alpha_0 \leq \alpha_0'$. By subtracting Equation (28) and Equation (36), we get

$$\alpha_0 N - \alpha_0' N = \frac{(\sqrt{A_1} - \sqrt{A_2})(G_2 - G_1)}{2(\sqrt{A_1A_2A_3} + \sqrt{A_0A_2A_3} + \sqrt{A_0A_1A_3} + \sqrt{A_0A_1A_2})}. \tag{37}$$

From the above equation, we know that the processing time is a minimum if and only if the sequence of load distribution based on ascending order of communication speed parameter, i.e., $G_1 \leq G_2$. Hence, from the above equation, we can change the arrangement if and only if the processing speed A_2 is less than A_1. Now, we generalize the result as follows:

Optimal Arrangement Theorem: Given an $(m + 1)$-processor single-level tree network with optimal sequence of load distribution, the total load processing time is minimum if the processors are connected to the links in ascending order of processor speed parameter A_i.

Proof: For m processors, consider a case when the root processor p_0 distributes the load fractions to child processors in the following sequence (p_1, p_2, $\cdots$, p_{i-1}, p_i, p_{i+1}, $\cdots$, p_m). Here, the network arrangement is (p_1, l_1), (p_2, l_2), $\cdots$, (p_i, l_i), (p_{i+1}, l_{i+1}), $\cdots$, (p_m, l_m). The value of load fraction α_0 assigned to the root processor in this arrangement is given as Equation (31).

Consider another arrangement where a processor p_i is connected to a link l_{i+1} and a processor p_{i+1} is connected to a link l_i, i.e., the arrangement is (p_1, p_2, $\cdots$, p_{i-1}, p_i, p_{i+1}, $\cdots$, p_m). Here, the network arrangement is (p_1, l_1), (p_2, l_2), $\cdots$, (p_{i+1}, l_i), (p_i, l_{i+1}), $\cdots$, (p_m, l_m). The value of load fractions assigned to the root processor in this arrangement is given as Equation (32).

The load fraction for the new arrangement can be obtained by exchanging the A_i and A_{i+1} in Equation (31). The interchange affects terms f_i, f_{i+1}, f_{i+2}, β_i and β_{i+1} only, and does not affect the other terms. Note that, because of this interchange, $y(m)$ and $y'(m)$ will not change. Now, we will find the conditions for $\alpha_0 \leq \alpha_0'$ which is the same as $x(m) \leq x'(m)$. The terms $x(m)$ and $x'(m)$ are a function of fs and βs.

Now, $x(m) - x'(m)$ is given as follows:

$$x(m) - x'(m) = \frac{(G_{i+1} - G_i)(\sqrt{A_i} - \sqrt{A_{i+1}}) \left\{\sum_{j=i+2}^{m} \prod_{k=i+2}^{j} \sqrt{f_k}\right\}}{2\sqrt{A_i}A_{i+1}}. \tag{38}$$

Then,

$$\alpha_0 N - \alpha_0' N = \frac{(G_{i+1} - G_i)(\sqrt{A_i} - \sqrt{A_{i+1}}) \left\{\sum_{j=i+2}^{m} \prod_{k=i+2}^{j} \sqrt{f_k}\right\}}{2y(m)\sqrt{A_i}A_{i+1}}. \tag{39}$$

Here, note that $\alpha_0 N \leq \alpha_0' N$ only when $A_i \leq A_{i+1}$. By recursively applying the above condition, we can get the optimal load distribution sequence which satisfies the condition $A_1 \leq A_2 \leq \cdots \leq A_m$. This proves the theorem.

In the above analysis, the speed condition of the root processor is not included. Now, we will prove the speed condition on the root processor.

Let us consider a two-processor network and the arrangement of processors in the network is (p_1, l_1) and (p_2, l_2). The processing time for this arrangement is

$$T = \left\{ \frac{2N\sqrt{A_1A_2} + G_1}{2\left(\sqrt{A_1A_2} + \sqrt{A_0A_1} + \sqrt{A_0A_2}\right)} \right\}^2 A_0. \tag{40}$$

Now, assume that the processor p_1 should distribute the load fractions instead of processor p_0. Then, we have to consider another arrangement: (p_0, l_1) and (p_2, l_2). The total load processing time for this arrangement is

$$T' = \left\{ \frac{2N\sqrt{A_0A_2} + G_1}{2\left(\sqrt{A_0A_2} + \sqrt{A_1A_0} + \sqrt{A_1A_2}\right)} \right\}^2 A_1. \tag{41}$$

The value $T - T'$ is computed as follows:

$$T - T' = \frac{G_1[4N\sqrt{A_0A_1A_2} + G_1(\sqrt{A_0} + \sqrt{A_1})]}{2\left(\sqrt{A_0A_1} + \sqrt{A_0A_2} + \sqrt{A_1A_2}\right)^2}(\sqrt{A_0} - \sqrt{A_1})..\tag{42}$$

Hence, $T \leq T'$ only when $A_0 \leq A_1$. From here, we can say that the first processor should be fastest. Note that, to find speed condition of the root processor, we have to use the processing time expression. For the speed condition of the child processors, it is sufficient to consider the value of the α_0 expression rather than the processing time expression.

Numerical Example 3: In this example, we consider the same parameters used in the Numerical Example 1. In the Numerical Example 1, we have used load distribution sequence (p_1, p_2, p_3). The total load processing time is 148,384 units. By applying the optimal arrangement theorem, the optimal sequence of load distribution is (p_2, l_3), (p_1, l_2), (p_0, l_1). The load originating processor is now p_3. The total load processing time is 147,975 units. From this result, we can see that the total processing time with the optimal sequence and arrangement is less than that of the total load processing time for the other sequences.

V. Conclusions

In this paper, we have dealt with parallel processing of second order computational loads in a single-level tree network with the non-blocking mode of communication. With a mild assumption on communication to computation speed ratio, we have shown how to derive a closed-form expression for optimal load partition such that the total load processing time is minimum. Numerical examples are presented to illustrate the closeness of the solution. The main advantage of the closed-form expression is in the study of characteristics of the system. Using the closed-form expressions, we derive the condition for optimal sequencing and arrangements of processors. These results can be used in intelligent scheduling of divisible second order processing loads.

Acknowledgment

The third author would like to acknowledge the support by the IT R&D program [2008-F-036-02, Development of anonymity-based u-knowledge security technology], and the CTRC program of MCST/KOCCA, Korea University, and the 3DLife project by the National Research Foundation. The fourth author would like to acknowledge the support of DOE grant DE-SC0003361. The authors would like to thank the reviewers for their valuable comments and suggestions which has improved the quality of presentation.

Appendix

For linear processing loads, it has been proved that the processing time is minimum only when all processors stop computing at the same time [8]. In this appendix, we will prove that it is true even for nonlinear computational loads. First, we present an motivational example and next we formally define the theorem and prove it.

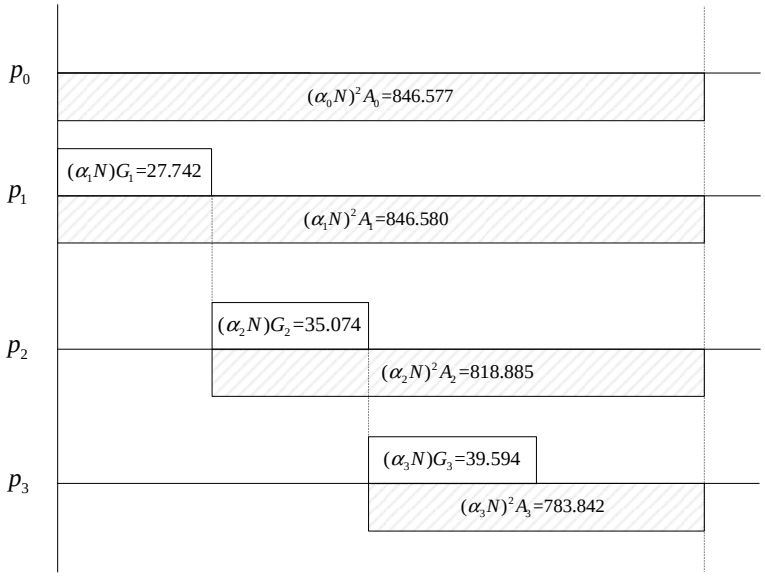

Fig. A1. Timing diagram for load distribution process ($m = 3$)

A. Numerical Example A1:

Let us consider a three-processor ($m = 3$) system with the following parameters: $A_0 = 1$, $A_1 = 1.1$, $A_2 = 1.5$, $A_3 = 2$, $G_1 = 1$, $G_2 = 1.5$, and $G_1 = 2$. Total size of the processing load is 100. First, we assume that the processors participating in the computation stop computing at the same time. Using our closed-form expression of the load fraction, we can determine the size of load fractions assigned to the processors. The load fractions are: $\alpha_0 = 0.29096$, $\alpha_1 = 0.27742$, $\alpha_2 = 0.23365$ and $\alpha_3 = 0.19797$. The timing diagram describing the communication and computation time for each processor is shown in Fig. A1.

From the timing diagram shown in Fig. A1, the finishing times for processors p_0, p_1, p_2 and p_3 are: $T_0 = 846.577$, $T_1 = 846.580$, $T_2 = 846.627$ and $T_3 = 846.631$. The total load processing time is the maximum of T_1, T_2, T_3, and T_4 which is 846.631. There is a small deviation in finishing times due to approximation in the derivation of the load fractions.

Since the child processor p_2 can compute faster than p_3, we assign additional load from p_3 to p_2. Now, the load fractions are $\alpha_0 = 0.29096$, $\alpha_1 = 0.27742$, $\alpha_2 = 0.24365$, and $\alpha_3 = 0.18797$. For this load distribution, the timing diagram is shown in Fig. A2. From the figure, the finishing times for processors p_0, p_1, p_2 and p_3 are: $T_0 = 846.577$, $T_1 = 846.580$, $T_2 = 918.19$ and $T_3 = 770.919$. From the result, we can see that the child processor p_2 requires more time to complete the load processing, where as others finish their computation earlier. The total load processing time is a maximum of T_1, T_2, T_3, and T_4 which is 918.19. From this result, we can say that the total processing time is the minimum if all participating processors stop computing at the same time. Now, we formally state the theorem for non-linear case and prove the statement is true.

Theorem I: If all nodes of the nonlinear computing model receiving non-zero load fractions stop computing at the same time, then the processing time T is a minimum.

Proof: Let $\alpha = \{\alpha_0, \alpha_1, \cdots, \alpha_m\}$ be the load fractions assigned to the processors $p_0, p_1, \cdots, p_m$ respectively. Let $T_0, T_1, \cdots, T_m$ be the corresponding finishing times.

Case A: We consider the finishing times of processor p_0 and p_1. The rest of the finishing times are assumed to be arbitrary and the load fractions assigned to other processors are assumed to be arbitrary constants.

$$C_0 = \sum_{i=2}^{m} \alpha_i. \tag{A1}$$

Here, C_0 is a constant. Then

$$\alpha_1 = 1 - \alpha_0 - \sum_{i=2}^{m} \alpha_i = (1 - C_0) - \alpha_0, \quad 0 \leq \alpha_0 \leq 1 - C_0. \tag{A2}$$

From the timing diagram given in Fig. A1, we can write the finishing times of processor p_0 and p_1 as

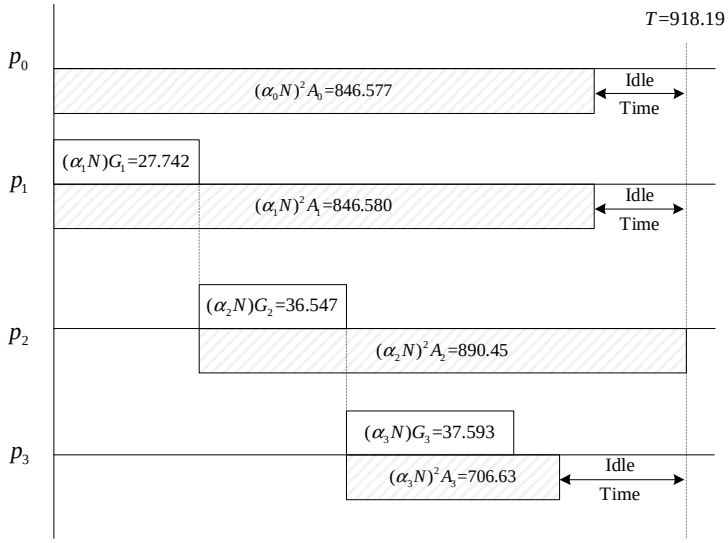

Fig. A2. Timing diagram for load distribution process $(m = 3)$ by changing the load fraction assigned to p_2.

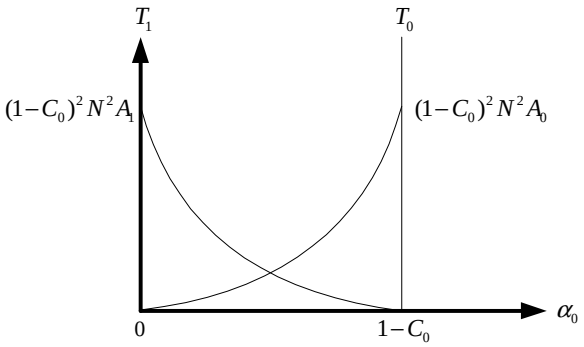

Fig. A3. Variation of finishing times for processor p_0 and p_1

$$\begin{aligned} T_0 &= (\alpha_0 N)^2 A_0, \\ T_1 &= (\alpha_1 N)^2 A_1. \end{aligned} \tag{A3}$$

By substituting α_1 in T_1, we get

$$T_1 = (1 - C_0 - \alpha_0)^2 N^2 A_1. \tag{A4}$$

The optimal processing time is the time that minimizes the $\max\{T_0, T_1\}$. The variation of finishing times T_0 and T_1 for different values of α_0 are given in Fig. A3.

From the Fig. A1, we can see that the processing time is a minimum, if the finishing times for processor p_0 and p_1 are the same, i.e., $T_0 = T_1$. At this point, we can express α_1

$$\alpha_1 = \alpha_0 \sqrt{\frac{A_0}{A_1}} = k_1 \alpha_0. \tag{A5}$$

Case B: Now, we examine the case with three processors (p_0, p_1, p_2) and their finishing times are T_0, T_1 and T_2, respectively. Here again we assume that the load fractions assigned to other processors in the network are arbitrary constants.

$$C_1 = \sum_{i=3}^{m} \alpha_i. \tag{A6}$$

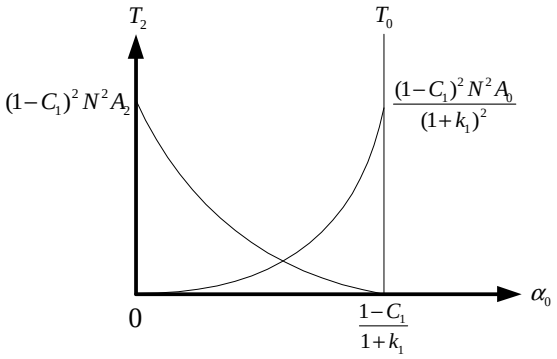

Fig. A4. Variation of finishing times with respect to loads fraction α_0

Now, the load fraction assigned to the child processor p_2 can be expressed in terms of load fraction α_0 and α_1 as,

$$\alpha_2 = 1 - (\alpha_3 + \alpha_4 + \cdots \alpha_m) - \alpha_0 - \alpha_1. \tag{A7}$$

Using Equations (A5) and (A6), we can express α_2 in terms of α_0 as

$$\alpha_2 = 1 - C_1 - (1 + k_1)\,\alpha_0, \quad 0 \leq \alpha_0 \leq \frac{1 - C_1}{1 + k_1}, \tag{A8}$$

where $k_1 = \sqrt{f_1}$. From the timing diagram given in Fig. A2, finishing time for T_2 and T_0 are expressed as

$$T_0 = (\alpha_0 N)^2 A_0. \tag{A9}$$

$$T_2 = (\alpha_1 N)\, G_1 + (\alpha_2 N)^2 A_2. \tag{A10}$$

The finishing time T_2 for processor p_2 can be expressed in terms of α_0 as

$$T_2 = (k_1 \alpha_0 N)\, G_1 + ([1 - C_1 - (1 + k_1)\,\alpha_0]\, N)^2 A_2. \tag{A11}$$

Now, we plot the finishing times T_0 and T_2 with respect to the load fraction α_0 as shown in Fig. A4. When the load fraction α_0 equals to the value $(1-C_1)/(1+k_1)$, the load fraction α_2 assigned to the processor p_2 is zero. Hence, the finishing time T_2 is zero. From the figure, we can observe that the finishing times meet each other at one point which is the minimum processing time point. From the previous case, we can say that the finishing time of T_1 is the same as T_0. Hence, at the minimum point, $T_2 = T_1 = T_0$.

Using this condition, we can express the load fraction α_2 in terms of the load fraction α_0 as given in Equation (18)

$$\alpha_2 N = \alpha_0 N \sqrt{f_1 f_2} - \frac{\beta_1 \sqrt{f_2}}{2} = k_2 \alpha_0 N - r_2, \tag{A12}$$

where $k_2 = \sqrt{f_1 f_2}$ and $r_2 = \beta_1 \sqrt{f_2}/2$.

Case C: Now, we examine four processors (p_0, p_1, p_2, p_3) and their finishing times are T_0, T_1, T_2 and T_3, respectively. Here again, we assume that the load fractions assigned to other processors in the network are arbitrary constants.

$$C_2 = \sum_{i=4}^{m} \alpha_i. \tag{A13}$$

Now, the load fraction assigned to the child processor p_3 can be expressed in terms of the load fraction α_0, α_1, and α_2 as,

$$\alpha_3 = 1 - (\alpha_4 + \alpha_5 + \cdots \alpha_m) - \alpha_0 - \alpha_1 - \alpha_2. \tag{A14}$$

Using Equations (A13), (A12) and (A5), we can express α_3 in terms of α_0 as

$$\alpha_3 = 1 - C_2 + \tfrac{r_2}{N} - (1 + k_1 + k_2)\alpha_0, 0 \leq \alpha_0 \leq \tfrac{(1-C_2+r_2/N)}{(1+k_1+k_2)}. \tag{A15}$$

From the timing diagram given in Fig. A2, finishing time for T_3 is expressed as

$$T_3 = (\alpha_1 N)\, G_1 + (\alpha_2 N)\, G_2 + (\alpha_3 N)^2 A_3. \tag{A16}$$

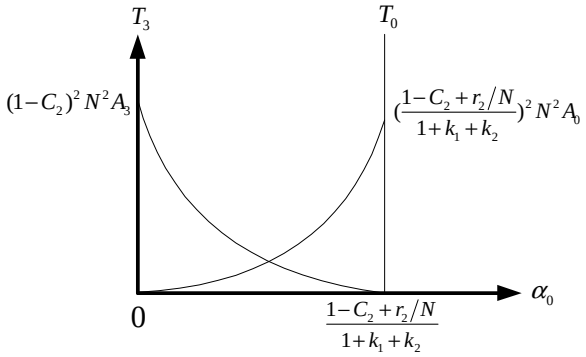

Fig. A5. Variation of finish times with respect to load fraction α_0.

The finishing time T_3 for processor p_3 can be expressed in terms of α_0 as

$$T_3 = (k_1\alpha_0 N)G_1 + (k_2\alpha_0 N - r_2)G_2 + \left(\left[1 - C_2 + \frac{r_2}{N} - (1 + k_1 + k_2)\alpha_0\right] N\right). \tag{A17}$$

Now, we plot the finishing times T_0 and T_3 which is shown in Fig. A5. When the load fraction α_0 equals to the value $(1 - C_1 + r_2/N)/(1 + k_1 + k_2)$, the load fraction α_3 assigned to processor p_3 is zero. Hence, the finishing time T_3 at this condition is zero. From the figure, we can observe that the finishing times meet each other at one point which is the minimum processing time point. From previous cases, we can say that the finishing times of T_1 and T_2 is the same as T_0. Hence, at the minimum point, $T_3 = T_2 = T_1 = T_0$.

Using this condition, we can express the load fraction α_3 in terms of load fraction α_0 as given in Equation (18),

$$\alpha_3 N = \alpha_0 N\sqrt{f_1 f_2 f_3} - \frac{\beta_1\sqrt{f_2 f_3}}{2} - \frac{\beta_2\sqrt{f_3}}{2} = k_3\alpha_0 N - r_3 \tag{A18}$$

where $k_3 = \sqrt{f_1 f_2 f_3}$ and $r_3 = \frac{\beta_1\sqrt{f_2 f_3}}{2} + \frac{\beta_2\sqrt{f_3}}{2}$

Case D: Based on the results in the previous cases, we can extend the proof to show that minimum processing time is achieved when $T_0 = T_1 = \cdots = T_i$ for $i + 1$ processors (p_0, p_1, $\cdots$, p_i). Let

$$C_i = \sum_{j=i+1}^{m} \alpha_j. \tag{A19}$$

Then,

$$\alpha_i \quad = \quad 1 - C_i - \sum_{j=0}^{i-1} \alpha_j \tag{A20}$$

From the results of previous cases, we can express α_j in terms of α_0 as

$$\alpha_j \quad = \quad k_j\alpha_0 N - r_j, \quad j = 1, 2, \cdots, i-1 \tag{A21}$$

where $k_j = \sqrt{\prod_{k=1}^{j} f_k}$ and $r_j = \sum_{k=1}^{j-1} \frac{\beta_k\sqrt{\prod_{l=k+1}^{j} f_l}}{2}$. Note that $r_1 = 0$.

Now, we can express α_i in terms of α_0 as

$$\alpha_i \quad = \quad 1 - C_i + \sum_{k=1}^{i-1} \frac{r_k}{N} - (k_1 + \cdots + k_{i-1})\,\alpha_0 \tag{A22}$$

From the above equation, the feasible values for α_0 are

$$0 \le \alpha_0 \le \frac{1 - C_i + \sum_{k=1}^{i-1} \frac{r_k}{N}}{(k_1 + \cdots + k_{i-1})} = C \tag{A23}$$

From the timing diagram given in Fig. 2, finish time T_i for processor p_i can be expressed as

$$T_i = (\alpha_1 N)\,G_1 + \cdots + (\alpha_{i-1} N)\,G_{i-1} + (\alpha_i N)^2\,A_i \tag{A24}$$

When $\alpha_0 = C$, the load fraction (α_i) assigned to the processor p_i is zero, and hence, finish time is zero. Similarly, when $\alpha_0 = 0$, the load fraction (α_i) assigned to the processor p_i is $1 - C_i$. Now, the finish time is $(1 - C_i)^2 NA_i$. From, this, we can conclude that there exist a minimum processing time at a crossover point where $T_0 = T_1 \cdots = T_i$. Using mathematical induction, one can generalize that the processing time is a minimum if all participating processors stop computing at the same time, i.e., $T_0 = T_1 = \cdots = T_m$.

REFERENCES

[1] Adler, M., Gong, Y., and Rosenberg, A. L., "Optimal sharing of bags of tasks in heterogeneous clusters," *Proceedings of the Annual ACM Symposium on Parallel algorithms and Architectures,* San Diego, California, USA, pp. 1-10, 2003.

[2] Bai, Y. and Robert, R. C., "Parallel block tridiagonalization of real symmetric matrices," *Journal of Parallel and Distributed Computing,*, vol. 68, pp. 703-715, 2008.

[3] Beaumont, O., Carter, L., Ferrante, J., Legrand, A., and Robert, Y., "Bandwidth-centric allocation of independent tasks on heterogeneous platforms," *Proceedings of the International Parallel and Distributed Processing Symposium*, IEEE Computer Society Press, 2002.

[4] Bharadwaj, V., Ghose, D. and Mani, V., "Optimal sequencing and arrangement in distributed single-level tree networks with communication delays," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 968-976, 1994.

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

[6] Beaumont, O., Casanova, H., Legrand, A., Robert, Y. and Yang, Y., "Scheduling divisible loadson star and tree networks: Results and open problems," *IEEE Transactions on Parallel Distributed Systems*, vol. 16, pp. 207-218, 2005.

[7] Beaumont, O., Legrand, A., and Robert, Y., "Scheduling divisible workloads on heterogeneous platforms," *Parallel Computing*, vol. 29, pp. 1121-1132, 2003.

[8] Bharadwaj, V., Ghose, D., and Mani, V., and Robertazzi, T. G., *Scheduling Divisible Loads in Parallel and Distributed Systems*, IEEE Computer Society, 1996.

[9] Bharadwaj V., Li X., and Ko C. C., "On the influence of start-up costs in scheduling divisible loads on bus networks," *IEEE Trans on Parallel and Distributed Systems*, vol. 11, no. 12, pp.1288-1305, 2000.

[10] Bharadwaj, V. and Viswanadham, N., "Suboptimal solutions using integer approximation techniques for scheduling divisible loads on distributed bus networks," *IEEE Transactions on System, Man, and Cybernetics-Part A: Systems and Humans,* vol. 30, pp. 680-691, 2000.

[11] Bataineh, S. and Robertazzi, T. G., "Distributed computation for a bus network with communication delays," *Proceedings of Information Science and Systems*, pp. 709-714, 1991.

[12] Bataineh, S. and Robertazzi, T. G., "Bus oriented load sharing for a network of sensor driven processors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 5, pp. 1202-1205, 1991.

[13] Cheng, Y. C. and Robertazzi, T. G., "Distributed computation with communication delays," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 24, no. 6, pp. 700-712, 1988.

[14] Cheng, Y. C. and Robertazzi, T. G., "Distributed computation for a tree network with communication delays," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 26, no. 3, pp. 511-516, 1990.

[15] Drozdowski, M., Lawenda, M., and Guinand, F., "Scheduling multiple divisible loads,"*The International Journal of High Performance Computing Applications*, vol. 20, pp. 1930, 2006.

[16] Drozdowski, M. and Lawenda, M., "The combinatorics in divisible load scheduling,"*Foundations of Computing and Decision Sciences*, vol. 30, pp. 297308, 2005.

[17] Drozdowski, M. and Wolniewicz, P., "Out-of-core divisible load processing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, 1048-1056, 2003.

[18] Drozdowski, M. and Wolniewicz, P., "Optimum divisible load scheduling on heterogeneous stars with limited memory,"*European Journal of Operational Research*, vol. 172, pp. 545559, 2006.

[19] Dutot, P.-F., "Divisible load on heterogeneous linear array," *Proceedings of the International Parallel and Distributed Processing Symposium*, Nice, France 2003.

[20] Ghose, D., Kim, H. J., and Kim, T. H., "Adaptive divisible load scheduling strategies for workstation clusters with unknown network resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 10, pp. 897-907, 2005.

[21] Hung, J. T., Kim, H. J., and Robertazzi, T. G., "Scalable scheduling in parallel processors," *Proceedings of the Conference on Information Sciences and Systems*, Princeton University, Princeton, NJ, 2002.

[22] Hung, J. T. and Robertazzi, T. G., "Distributed Scheduling of Nonlinear Computational Loads," *Proceedings of the Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore MD, March 2003.

[23] Hung, J. T. and Robertazzi, T. G., "Divisible load cut through switching in sequential tree networks,"*IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, pp. 968-982, 2004.

[24] Khalifa, K. B., Boubaker, M., Chelbi, N., and Bedoui, M. H., "Learning vector quantization neural network implementation using parallel and serial arithmetic," *International Journal of Computer Sciences and Engineering Systems*, vol. 2, No. 4, pp. 251-256, 2008.

[25] Kim, H. J., "A novel optimal load distribution algorithm for divisible loads," *Cluster Computing*, vol. 6, no. 1, pp. 41-46, 2003.

[26] Kim, H. J., Jee, G.-I., and Lee, J. G., "Optimal load distribution for tree network processors," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 32, no. 2, pp. 607-612, 1996.

[27] Orr, R. S., "The order of computation for finite discrete Gobar transform," *IEEE Transactions on Signal Processing*, vol. 41, No. 1, pp. 122-130, 1993.

[28] Othman, H. and Aboulnasr, T., "A separable low complexity 2D HMM with application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1229-1238, 2003.

[29] Piriyakumar, D. A. L., and Murthy, C. S. R., "Distributed computation for a hypercube network of sensor-driven processors with communication delays including setup time," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 28, pp. 245-251, 1998.

[30] Robertazzi, T. G., "Processor equivalence for daisy chain load sharing processors," *EEE Transactions on Aerospace and Electronics Systems*, vol. 29, no. 4, pp. 1216-1221, 1993.

[31] Robertazzi, T. G., "Ten reasons to use divisible load theory," *IEEE Computer*, vol. 36, no. 5, pp. 63-68, 2003.

[32] Sohn J., and Robertazzi, T. G., "Optimal divisible job load sharing on bus networks," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 32, no. 1, pp. 34-40, 1996.

[33] Special Issue, "Divisible load scheduling," *Cluster Computing*, vol. 6, no. 1, pp. 5-86, 2003.

[34] Suresh, S., Kim, H. J., Cui, R., and Robertazzi, T. R., "An analytical solution for scheduling nonlinear divisible loads for a single level tree network with a collective communication model," Technical Report. 001, Nanyang Technological University, 2009.

[35] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J., "Divisible load scheduling in distributed system with buffer constraints: Genetic algorithm and linear programming approach," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 21, no. 5, pp. 303-321, 2006.

[36] Suresh, S., Omkar, S. N., and Mani, V., "The effect of start-up delays in scheduling divisible loads on bus networks: An alternate approach," *Computer and Mathematics with Applications*, vol. 46, no. 10-11, pp. 1545-1557, 2003.

[37] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J., "An equivalent network for divisible load scheduling in nonblocking mode of communication," *Computers and Mathematics with Applications*, vol. 49, no. 9-10, pp. 1413-1431, 2005.

[38] Suresh, S., Mani, V., Omkar, S. N., Kim, H.J., and Sundararajan, N."A new load distribution strategy for linear network with communication delays," *Mathematics and Computers in Simulation*, vol. 79, no. 5, pp. 1488-1501, 2009.

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

[40] Duda, R. O. and Hart, P. E., "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, pp. 11-15, 1972.
[41] Dennis, J. E. Jr., "Nonlinear Least-Squares," *State of the Art in Numerical Analysis*, ed. D. Jacobs, Academic Press, pp. 269-312. 1977.
[42] Guil, N., Villalba, J., and Zapata, E. L., "A fast Hough transform for segment detection," *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1541-1548, 1995.