Divisible Load Scheduling in Single-Level Tree Networks: Optimal Sequencing and Arrangement in the Nonblocking Mode of Communication

H. J. Kim*
Department of Control and Instrumentation Engineering
Kangwon National University, Chunchon 200-701, Korea
khj@kangwon.ac.kr

V. Mani
Department of Aerospace Engineering, Indian Institute of Science
Bangalore, 560 012 India

(Received May 2002; revised and accepted November 2002)

Abstract—The problem of scheduling divisible loads in a single-level tree network is considered. All the earlier studies in divisible load scheduling consider the blocking mode of communication. In this paper, the nonblocking mode of communication is considered. Closed-form expression for the processing time in the nonblocking mode of communication is derived. Using this closed-form expression, this paper analytically proves a number of significant results on load sequencing and processor-link arrangement in a single-level tree network. It is shown that the performance measures of interest such as the processing time, idle time, and utilization of processors are better in the nonblocking mode of communication than in the blocking mode of communication. © 2003 Elsevier Ltd. All rights reserved.

Keywords—Divisible loads, Nonblocking mode of communication. Processing time, Optimal load sequence, Single-level tree network.

1. INTRODUCTION

The main objective in parallel and distributed computing systems is the minimization of processing time of the jobs that arrive at any of the computer sites in the system. Minimizing the processing time involves designing efficient scheduling algorithms. An efficient scheduling algorithm distributes the jobs in an optimal manner to the set of available processors in the system so that the processing time is minimum [1–3]. In the context of scheduling indivisible and modularly divisible loads, there is a large amount of research results available in the literature. The domain of scheduling divisible loads in multiprocessor systems started in 1988 and has stimulated considerable interest among researchers and engineers [4]. An introduction to divisible load scheduling

*Author to whom all correspondence should be addressed.

This work was in part supported by the Advanced Information Technology Research Center (AITRC), KAIST, under the auspices of the Ministry of Science and Technology, Korea.
This paper was in part supported by the IT Research Center (ITRC), Kangwon National University.
in distributed computing systems and recent research studies in this area are presented in [4]. 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.

**Practical Example of Divisible Load.** (See [5].) Consider the automated fingerprint matching application. Here, the problem is as follows: the unknown fingerprint ("a mark") must be compared against a large number of possible matches. A score (reflecting goodness of fit) is computed, and the top few candidates are then provided (in sorted order from the best to the worst) to a human expert for final verification. For a given set of possible matches, the problem is that of obtaining (computing) a score for each of them. The set of possible matches is the divisible load and can be divided into any number of fractions and the score can be computed independently on the processors in a distributed computing system. This type of problem without communication delay (time delay in sending the possible matches to processors in the system) is called "embarrassingly parallel" in [5]. When there is no communication delay, it is shown in [5] that, if there are \( n \) identical processors available in the system, then each could take \( 1/n \) of the large possible matches and process them in parallel. Note that the number of fingerprints constituting the total load is very large compared to the number of processors. Due to this, the load can be considered as arbitrarily divisible, i.e., a continuous entity. A few of the other application domains of scheduling divisible loads are image processing applications using Hough transform [6], matrix-vector products of very large sizes [7], computer vision data processing [8], query processing in database systems [9], distributed biomedical image processing [10], and filtering, encryption, and coding problems [11].

The problem of scheduling divisible loads in a linear network incorporating the associated communication delay was first introduced in [12]. In this study [12], the load distribution process is represented as a Gantt-chart-like timing diagram, and the recursive load distribution equations are introduced. The methodology from this paper is extended to scheduling divisible loads in tree networks in [13] and to bus networks in [14]. In these studies, the optimal load fractions are obtained by assuming that all the processors involved in the computation of the processing load stop computing at the same time instant. This assumption has been shown to be a necessary and sufficient condition to obtain optimal processing time for linear networks in [15] and bus networks in [16]. However, it has been rigorously shown that this condition is true only in a restricted sense [17], in the case of a heterogeneous single-level tree network. A closed-form expression for the processing time is presented in [18,19] for a single-level tree network, and using this closed-form expression, optimal sequence of load distribution and optimal arrangement of processors and links are obtained in [18]. For the case of homogeneous linear and tree networks, their closed-form expressions and an asymptotic performance analysis are presented in [20,21]. Scheduling divisible loads in hypercube, mesh networks and other associated issues are discussed in [22–24].

In all these studies [7,9,12–24] on divisible load scheduling, the process of communicating the load fractions is the "blocking mode of communication". In the blocking mode of communication, the processor will start the computation process only after the front-end (communication co-processor) has received all the load fractions assigned to that processor. Hence, there is a delay in starting the computation process by the processor. This delay cannot be removed completely for all processors in the network, but can be reduced in another mode of communication known as the "nonblocking mode of communication". In the nonblocking mode of communication, the processor will start the computation process, while its front-end is receiving the processing load. Hence, the delay in starting the computation process is reduced in the nonblocking mode of communication in comparison with the blocking mode of communication. In the fingerprint example mentioned above, consider that a processor is assigned with 100 possible matches to compute the score on each of these matches. In the blocking mode of communication, this processor will start the computation process only after receiving all the 100 possible matches.
In the nonblocking mode of communication, this processor will start the computation process after receiving the first possible match of the 100 possible matches. This nonblocking mode of communication in divisible load scheduling was first introduced in [11] for a homogenous tree network.

In this paper, scheduling divisible loads in the nonblocking mode of communication and in a heterogeneous single-level tree network are considered. We first derive the load distribution equations for this nonblocking mode of communication and obtain a closed-form expression for the processing time in this mode. Using this closed-form expression, we present the important results on an optimal sequence of load distribution and optimal arrangement of links and processors in this tree network. We also discuss all issues related to this mode of communication with the earlier studies.

2. DEFINITIONS AND PROBLEM FORMULATION

A single-level tree network with \((m+1)\) processors considered in this paper is shown in Figure 1. All the processors are connected to the root processor \(p_0\) via communication links. This tree configuration can be represented as an ordered set as follows:

\[
T(p_0) = \{(l_1, p_1), (l_2, p_2), \ldots, (l_1, p_1), \ldots, (l_m, p_m)\},
\]

where \((l_i, p_i)\) represents the \(i^{th}\) child processor \(p_i\) and is connected to the root processor \(p_0\) via link \(l_i\). This ordered set represents the arrangement of \(m + 1\) processors and \(m\) links. The order also represents the sequence in which the root processor distributes the processing load to other processors. The divisible load arrives at the root processor \(p_0\). The root processor \(p_0\) divides the processing load into \((m + 1)\) fractions \((\alpha_0, \alpha_1, \ldots, \alpha_m)\) and keeps the part \((\alpha_0)\) for itself to process/compute and distributes fractions \((\alpha_1, \alpha_2, \ldots, \alpha_m)\) to the child processors in the sequence \(p_1, p_2, \ldots, p_m\) one after another. The processors in the network are equipped with front-ends (or communication co-processors). The child processors start computing the load fractions while the front-end is receiving the load fractions. The objective here is to find the optimal size of these load fractions \(\alpha_0, \alpha_1, \ldots, \alpha_m\) such that the processing time is minimum.

![Figure 1. A single-level tree network with \((m + 1)\) processors.](image)

Definitions

**LOAD DISTRIBUTION.** This is denoted as \(\alpha\) and defined as an \((m + 1)\)-tuple \((\alpha_0, \alpha_1, \ldots, \alpha_m)\) such that \(0 \leq \alpha_i \leq 1\). The equation \(\sum_{i=0}^{m} \alpha_i = 1\) is the normalization equation, and the space of all load distribution is denoted as \(\Gamma\).

**FINISH TIME.** The finish time of processor \(p_i\) is denoted as \(T_i\) and is defined as the time difference between the instant at which the \(i^{th}\) processor stops computing and the time instant at which the root processor initiates the process.
Processing Time. This is denoted as $\Gamma(T(p_0))$ and defined as follows:

$$\Gamma(T(p_0)) = \max(T_0, T_1, \ldots, T_m),$$  
(2)

where $T_i$ is the finish time of processor $p_i$.

Optimal Load Distribution. This is defined as the load distribution for a given arrangement and a given sequence such that $\Gamma(T(p_0))$ is minimum.

Optimal Sequence. This is defined as that sequence of optimal load distribution for a given arrangement such that $\Gamma(T(p_0))$ is minimum.

Optimal Arrangement. This is defined as the arrangement of links and processors such that $\Gamma(T(p_0))$ is minimum, provided the optimal sequence and optimal load distribution is followed.

Notations

- $Q_i$: Fraction of the processing load assigned to processor $p_i$
- $W_i$: A constant inversely proportional to the speed of the processor $p_i$
- $Z_i$: A constant inversely proportional to the speed of the link $l_i$
- $T_{cm}$: Time to communicate the entire processing load by a standard processor
- $T_{cp}$: Time to compute/process the entire processing load by a standard processor
- $\sigma$: Ratio of communication time to processing time for a given load in a standard link and processor (i.e., $\sigma = T_{cm}/T_{cp}$)

For a standard processor and standard link, let $w_i = 1$ and $z_i = 1$, respectively. In divisible load theory literature almost all researchers follow these definitions and notations. In [17], it has been rigorously proved that for the optimal processing time all the processors involved in the computation of the processing load must stop computing at the same time instant. We, in this paper also, use this optimality criterion.

Closed-Form Expression for Processing Time

Now, we derive the closed-form expression for the processing time. This is derived by assuming that the sequence of load distribution is from $p_1, p_2, \ldots, p_m$ in that order. This means that the root processor divides the processing load into $m + 1$ fractions, namely, $\alpha_0, \alpha_1, \ldots, \alpha_m$, and keeps the fraction $\alpha_0$ for itself for processing. It transmits the remaining fractions $\alpha_1, \alpha_2, \ldots, \alpha_m$ to the processors $p_1, p_2, \ldots, p_m$, respectively, one after another. So, the sequence of load distribution is $p_1, p_2, \ldots, p_m$. The timing diagram is the usual way of representing the load distribution process in divisible load scheduling. The timing diagram (for the nonblocking mode of communication) for optimal load distribution is shown in Figure 2. Here, it is the nonblocking mode of communication. This means that the child processor will start the computation process while its front-end is receiving the processing load. The timing diagram (for the blocking mode of communication) for optimal load distribution is given in [18]. Note that, in this timing diagram of Figure 2, the first child processor starts computing at the same time as the root processor. Also note that, in this timing diagram, all the processors involved in the computation process stop computing at the same time instant. From the timing diagram shown in Figure 2, the recursive load distribution equations are obtained as

$$\alpha_0 w_0 T_{cp} = \alpha_1 w_1 T_{cp},$$

$$\alpha_i w_i T_{cp} = \alpha_i z_i T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}, \quad i = 1, 2, \ldots, m - 1,$$  
(3)

and the normalization equation is given as follows:

$$\sum_{j=0}^{m} \alpha_j = 1.$$  
(4)
Equation (3) can be rewritten as follows:

\[ \alpha_i = f_{i+1} \alpha_{i+1}, \quad i = 0, 1, \ldots, m - 1, \tag{5} \]

where

\[ f_j = \frac{w_j T_{cp}}{w_j T_{cp} - z_j T_{cm}}, \quad j = 1, 2, \ldots, m. \tag{6} \]

Note here that \( z_0 \) is zero. It is also assumed that \( w_j T_{cp} > z_j T_{cm} \), for all \( j = 1, 2, \ldots, m \). The reason for this assumption is discussed in remarks at the end of this section. These recursive equations can be solved by expressing all the \( \alpha_i \) (\( i = 0, 1, \ldots, m - 1 \)) in terms of \( \alpha_m \) as follows:

\[ \alpha_i = \left\{ \prod_{j=i+1}^{m} f_j \right\} \alpha_m. \tag{7} \]

From the normalizing equation (4), the value of \( \alpha_m \) is obtained as

\[ \alpha_m = \frac{1}{1 + \sum_{k=1}^{m} \prod_{j=k}^{m} f_j}. \tag{8} \]

The load fraction assigned to processor \( p_i \) is \( \alpha_i \) and is

\[ \alpha_i = \frac{\prod_{j=i+1}^{m} f_j}{1 + \sum_{k=1}^{m} \prod_{j=k}^{m} f_j} \alpha_m, \quad i = 0, 1, \ldots, m - 1. \tag{9} \]

From the timing diagram in Figure 2, it can be seen that the processing time \( \Gamma(T(p_0)) \) is the processing time of the root processor \( p_0 \) given by \( \alpha_0 w_0 T_{cp} \). Thus, \( \Gamma(T(p_0)) \) is obtained as

\[ \Gamma(T(p_0)) = \frac{\left( \prod_{j=1}^{m} f_j \right) w_0 T_{cp}}{1 + \sum_{k=1}^{m} \prod_{j=k}^{m} f_j}. \tag{10} \]
This above closed-form expression for the processing time will be used to prove some important results on the optimal sequence of load distribution, and the optimal arrangement of links and processors in a single-level tree network with the nonblocking mode of communication.

Remarks

On the assumption $w_jT_{cp} > z_jT_{cm}$, $j = 1, 2, \ldots, m$, we note that one of the main reasons for using parallel and distributed computing is to share the heavy computational work among the processors because of the availability of a faster communication channel. Also, in practical problems, the computation time to obtain performance measures of interest on a particular data set by a processor will be much more than the time to communicate the data set to that processor. In the fingerprint matching examples presented earlier [5], the most time-consuming and difficult part of the process is the computation of the goodness of fit score. Communication time in sending the possible matches to the processors is very much less in comparison with the computation of the goodness of fit score. Here, $w_jT_{cp}$ is the time to process a given load fraction and $z_jT_{cm}$ is the time to communicate the given load fraction by the root processor. If this condition is not true, then that particular processor will not be computing continuously (perhaps through compute-wait cycle). Hence, in this paper, this assumption is used.

Another assumption in this paper is that we have a large number of possible matches assigned to a processor to compute the goodness of fit score. While deriving the load distribution equations (3) and (4), the communication time of the first possible match is ignored since the time to communicate one possible match is very small and can be ignored. The load distribution equations are derived with the idea that while the processor is computing the goodness of fit on the first data point, the front-end is receiving the other data points assigned to it. After computing the goodness of fit first data point, the processor starts the computation with the second data point, and so on.

3. OPTIMAL SEQUENCING AND ARRANGEMENT

In this section, we use the closed-form expression for the processing time given in the earlier section to prove important results on sequencing and arrangement. For this purpose, we use the load distribution pattern between two adjacent processor-link pairs and rewrite the closed-form expression in such a manner that only the terms corresponding to the $i^{th}$ and $(i+1)^{th}$ processor and link pairs are present explicitly, as given below:

$$
\Gamma(T(p_0)) = \frac{\left\{ \prod_{j=1, j \neq i, i+1, i+2}^m f_j \right\} f_{i+2}f_{i+1}f_i}{1 + \sum_{k=i+3}^m \prod_{j=k}^m f_j} + \left\{ \prod_{j=i+3}^m f_j \right\} \left( f_{i+2}f_{i+2}f_{i+1} + \sum_{k=i+1}^i \prod_{j=k, j \neq i+1, i+2}^m f_j \right) f_{i+2}f_{i+1}f_i - w_0T_{cp}, \quad (11)
$$

where

$$C(i) = \prod_{j=1, j \neq i, i+1, i+2}^m f_j, \quad (12)$$

$$K_1(i) = 1 + \sum_{k=i+3}^m \prod_{j=k}^m f_j, \quad (13)$$

$$K_2(i) = \prod_{j=i+3}^m f_j, \quad (14)$$

$$K_3(i) = \sum_{k=1}^i \prod_{j=k, j \neq i+1, i+2}^m f_j. \quad (15)$$
Divisible Load Scheduling

\[ K_3(i) \text{ can also be written as } (f_1 f_2 \cdots f_{i-1} f_{i+3} \cdots f_i) + (f_2 \cdots f_{i-1} f_{i+3} \cdots f_i) + \cdots + (f_{i-1} f_i) + f_i. \]

The reason for writing the processing time in this manner is that if any change in sequence of load distribution or arrangement of processors and links is done in these two processor-link pairs \( i \) and \( i + 1 \), then only \( f_i, f_{i+1}, f_{i+2} \) will change in the processing time expression. All other terms will not change and will be absorbed as constants defined above. Also, note that these expressions and constants are valid for only \( i = 1, 2, \ldots, m \). For the last two adjacent processor-link pairs \( m - 1 \) and \( m \), the corresponding expressions are

\[ T_m = (u_{m-1}, p_{m-1}) \]

\[ \Gamma(T(p_0)) = \frac{D(i) f_{m-1} f_m w_0 T_{cp}}{1 + K_4(i) f_{m-1} f_m + f_m}. \]

\[ D(i) = \prod_{j=1}^{m-2} f_j, \]

\[ K_4(i) = 1 + \sum_{k=1}^{i-1} \prod_{j=k}^{i-1} f_j. \]

Here, in \( K_4(i) \), \( i \) can take values only \( m - 1 \) and \( m - 2 \).

**Lemma 1.** For the nonblocking mode of communication in a single-level tree network \( T(p_0) \), if \( z_{i+1} \leq z_i \), and given that \( w_{i} T_{cp} > z_{i} T_{cm} \), and \( w_{i+1} T_{cp} > z_{i+1} T_{cm} \), for any two adjacent processor-link pairs, then the processing time will decrease or remain the same when \( (l_i, p_i) \) and \( (l_{i+1}, p_{i+1}) \) are interchanged.

**Proof.** When the pairs \( (l_i, p_i) \) and \( (l_{i+1}, p_{i+1}) \) are interchanged, the resulting arrangement \( T_A(p_0) \) is given as follows:

\[ T_A(p_0) = \{(l_1, p_1), (l_2, p_2), \ldots, (l_{i+1}, p_{i+1}), (l_i, p_i), \ldots, (l_m, p_m)\}. \]

We have to prove that \( \Gamma(T_A(p_0)) \leq \Gamma(T(p_0)) \), if \( z_{i+1} \leq z_i \). Because of this interchange only three terms \( f_i, f_{i+1}, \) and \( f_{i+2} \) in \( \Gamma(T(p_0)) \) will change. The changed values of \( f_{i}, f_{i+1}, \) and \( f_{i+2} \) are denoted as \( g_{i}, g_{i+1}, \) and \( g_{i+2} \), respectively, in the processing time expression of the changed sequence of load distribution. Thus, the processing time expression for the changed sequence of load distribution is \( \Gamma(T_A(p_0)) \) which is given as follows:

\[ \Gamma(T_A(p_0)) = \frac{C(i) g_{i+2} g_{i+1} g_i w_0 T_{cp}}{K_1(i) + K_2(i) (g_{i+2} + g_{i+2} g_{i+1}) + K_3(i) g_{i+2} g_{i+1} g_i}. \]

For clarity, we write \( f_i, f_{i+1}, \) and \( f_{i+2} \) and \( g_i, g_{i+1}, \) and \( g_{i+2} \) in terms of corresponding \( w_i \) and \( z_i \),

\[
\begin{align*}
  f_i &= \frac{w_i}{w_i - z_i}, & g_i &= \frac{w_i}{w_i - z_i}\,^\sigma, \\
  f_{i+1} &= \frac{w_{i+1}}{w_{i+1} - z_i}, & g_{i+1} &= \frac{w_i}{w_{i+1} - z_{i+1}}\,^\sigma, \\
  f_{i+2} &= \frac{w_{i+2}}{w_{i+1} - z_{i+1}}, & g_{i+2} &= \frac{w_{i+2}}{w_{i+1} - z_i}\,^\sigma, 
\end{align*}
\]

where \( \sigma = T_{cm}/T_{cp} \). The processing times of \( \Gamma(T(p_0)) \) and \( \Gamma(T_A(p_0)) \) are given in equations (11) and (20). Let us denote the numerators of equations (11) and (20) as \( N_1 \) and \( N_2 \), and respective denominators as \( D_1 \) and \( D_2 \). It can be easily seen, from equation (21), that \( f_{i+2} f_{i+1} f_i = g_{i+2} g_{i+1} g_i \), and hence, \( N_1 = N_2 \). We compare only the denominators \( D_1 \) and \( D_2 \) and obtain the following conditions:

- if \( D_1 - D_2 \geq 0 \), then \( \Gamma(T(p_0)) \leq \Gamma(T_A(p_0)) \),
- if \( D_1 - D_2 \leq 0 \), then \( \Gamma(T(p_0)) \geq \Gamma(T_A(p_0)) \).
The value of \( (D_1 - D_2) \) is obtained as

\[
D_1 - D_2 = \frac{K_2(i)w_{i+2}z_{i+1} - z_i}{(w_{i+1} - z_{i+1})s_i(z_i - z_{i+1})}.
\]

When \( z_{i+1} \leq z_i \), then \( (D_1 - D_2) \leq 0 \), which proves the lemma. Also, note that \( z_{i+1} < z_i \) means \( (D_1 - D_2) < 0 \), which implies a definite decrease in the processing time. This lemma is proved here for \( i = 1, 2, \ldots, m - 2 \). For the case \( i = m - 1 \), it can be proved in a similar manner using equations (16)-(18).

REMARKS. This lemma is valid only when \( w_iT_{cp} > z_iT_{cm}, w_{i+1}T_{cp} > z_{i+1}T_{cm}, \) and \( w_{i+1}T_{cp} > z_iT_{cm} \). As mentioned in the earlier studies [17], interchanging adjacent processor-link pairs do not imply a physical rearrangement in the architecture, but rather a change in the sequence of load distribution by the root processor.

THEOREM 1 (OPTIMAL SEQUENCE). In a single-level tree network \( T(p_0) \), in order to achieve minimum processing time, the sequence of load distribution in the nonblocking mode of communication by the root processor \( p_0 \) should follow the order in which the link speeds decrease.

PROOF. The proof directly follows from Lemma 1.

REMARKS. As mentioned in the earlier remarks, Theorem 1 is valid only when \( w_iT_{cp} > z_iT_{cm}, \) for all \( i = 1, 2, \ldots, m \).

LEMMA 2. For the nonblocking mode of communication, in a single-level tree network \( T(p_0) \), the following conditions exist.

1. If \( w_{i+1}T_{cp} > z_{i+1}T_{cm}, w_{i+1}T_{cp} > z_iT_{cm}, w_iT_{cp} > z_{i+1}T_{cm}, z_{i+1} > z_i \) and \( w_{i+1} \leq w_i \), for any two adjacent processor-link pairs \((l_i, p_i)\) and \((l_{i+1}, p_{i+1})\), then the processing time will decrease or remain the same when only the processors \( p_i \) and \( p_{i+1} \) are interchanged.

2. If \( z_{i+1} = z_i \) in the above condition, then the processing time is independent of the order in which the processors \( p_i \) and \( p_{i+1} \) are arranged.

PROOF.

1. When the processors \( p_i \) and \( p_{i+1} \) are interchanged, the resulting arrangement \( T_B(p_0) \) is as follows:

\[
T_B(p_0) = \{(l_1, p_1), (l_2, p_2), \ldots, (l_i, p_{i+1}), (l_{i+1}, p_i), \ldots, (l_m, p_m)\}. \tag{23}
\]

We have to prove that \( \Gamma(T_B(p_0)) \leq \Gamma(T(p_0)) \) if \( w_{i+1} < w_i \) along with the conditions given above. As mentioned in the proof of Lemma 1, only the terms \( f_i, f_{i+1} \), and \( f_{i+2} \) in \( \Gamma(T(p_0)) \) will change. The changed values are denoted as \( g_i, g_{i+1}, \) and \( g_{i+2} \). The processing time \( \Gamma(T_B(p_0)) \) is obtained as

\[
\Gamma(T_B(p_0)) = C(i)g_i + 2g_{i+1}g_i - g_i + 2g_{i+2}g_{i+1} + K_3(i)g_i + g_{i+2}g_{i+1}g_{i+2} - w_0T_{cp}. \tag{24}
\]

The values of \( g_i, g_{i+1}, \) and \( g_{i+2} \) are

\[
g_i = \frac{w_{i+1}}{w_{i+1} - z_{i+1}} \quad g_{i+1} = \frac{w_i}{w_{i+1} - z_i} \quad g_{i+2} = \frac{w_{i+2}}{w_{i'} - z_{i+1}}. \tag{25}
\]

The processing time for \( T(p_0) \) and \( T_B(p_0) \) are given in equations (11) and (24). Following the same procedure as in Lemma 1, the value of \( N_1D_2 - N_2D_1 \) is obtained as

\[
N_1D_2 - N_2D_1 = \frac{C(i)K_1(i)w_{i+1}w_{i+2}(w_i - w_{i+1})(z_{i+1} - z_i)s_iw_0T_{cp}}{(w_{i+1} - z_{i+1})s_i(w_{i+1} - z_{i+1})(w_i - z_i)(w_i - z_{i+1})}, \tag{26}
\]

if \( N_1D_2 - N_2D_1 \geq 0 \), then \( \Gamma(T(p_0)) > \Gamma(T_B(p_0)) \).

if \( N_1D_2 - N_2D_1 \leq 0 \), then \( \Gamma(T(p_0)) \leq \Gamma(T_B(p_0)) \).
Divisible Load Scheduling

From equation (26), we can see that if \( w_{i+1} \leq w_i \), and given that \( z_{i+1} > z_i \), then \( N_1D_2 - N_2D_1 \geq 0 \), which implies that \( \Gamma(T(p_0)) \geq \Gamma(T_B(p_0)) \). Hence, if \( w_{i+1} \leq w_i \), then by interchanging the processors \( p_i \) and \( p_{i+1} \), the processing time will decrease or remain the same if \( w_{i+1} < w_i \), which implies a definite decrease in the processing time and thus proving the first part of this lemma.

2. To prove the second part, we use equation (26). When \( z_{i+1} = z_i \), then \( N_1D_2 - N_2D_1 = 0 \), regardless of the value of \( w_i \) and \( w_{i+1} \), implying that the processing time is independent of the order in which the processors \( p_i \) and \( p_{i+1} \) are arranged.

Here too, this lemma is proved for \( i = 1, 2, \ldots, m - 2 \). For \( i = m - 1 \), it can proved in a similar way using equations (16)-(18).

REMARKS. Note, the values of \( g_i, g_{i+1} \), and \( g_{i+2} \) given in equation (25) used for proving this lemma are different from the values of \( g_i, g_{i+1} \) and \( g_{i+2} \) given in equation (21). Interchanging two adjacent processors implies a physical rearrangement in the architecture.

THEOREM 2. For the nonblocking mode of communication, in a single-level tree \( \Gamma(T(p_0)) \), if all the links have the speed \( z_i = z \), and \( w_iT_{cp} > zT_{cm} \), for \( i = 1, 2, \ldots, m \), then the processing time is independent of the order in which the processors are arranged.

PROOF. This proof directly follows from Lemma 2.

THEOREM 3. For the nonblocking mode of communication, in a single-level tree \( \Gamma(T(p_0)) \), if \( w_iT_{cp} > z_jT_{cm} \) for all \( i, j = 1, 2, \ldots, m \) in order to achieve minimum processing time the processors and links should be arranged in such a way that \( w_{i+1} \geq w_i \) and \( z_{i+1} \geq z_i \), for all \( i = 1, 2, \ldots, m \).

PROOF. The proof follows directly from Lemma 2.

REMARKS. This theorem shows how to achieve the minimum processing time when an architectural rearrangement of links and processors are possible. So far, the speed parameter \( w_0 \) of the root processor was not considered in the rearrangement. Now, we discuss this aspect.

LEMMA 3. For the nonblocking mode of communication in a single-level tree \( \Gamma(T(p_0)) \), if \( w_iT_{cp} > z_jT_{cm} \) for all \( i, j = 0, 1, \ldots, m \), with \( w_{i+1} \geq w_i \) and \( z_{i+1} \geq z_i \), for all \( i = 0, 1, \ldots, m - 1 \); if \( w_1 \leq w_0 \), then the processing time will decrease or remain the same by interchanging the root processor \( p_0 \) with the first left-hand side processor \( p_1 \).

PROOF. The fastest link-processor \( (l_1, p_1) \) pair will be in the first left position, because the processors and links are arranged in the manner \( w_{i+1} \geq w_i \) and \( z_{i+1} \geq z_i \). We now interchange the root processor \( p_0 \) with the first processor \( p_1 \). With this interchange the resulting arrangement \( T_C(p_1) \) is as follows:

\[
T_C(p_1) = \{(l_1, p_0), (l_2, p_2), \ldots, (l_m, p_m)\}. \tag{27}
\]

We have to prove that \( \Gamma(T_C(p_1)) \leq \Gamma(T(p_0)) \), if \( w_1 \leq w_0 \). The processing time for \( \Gamma(T(p_0)) \) and \( \Gamma(T_C(p_1)) \) where \( i = 0 \) are given as

\[
\Gamma(T(p_0)) = \frac{C(0)f_2f_1w_0T_{cp}}{K_1(0) + K_2(0)\{f_2 + f_2f_1\}}, \tag{28}
\]

\[
\Gamma(T_C(p_1)) = \frac{C(0)g_2g_1w_1T_{cp}}{K_1(0) + K_2(0)\{g_2 + g_2g_1\}}. \tag{29}
\]

For clarity, we write \( f_1, f_2 \) and \( g_1, g_2 \) in terms of \( w_i \) and \( z_i \) as

\[
f_1 = \frac{w_1}{w_0 - z_0\sigma}, \quad g_1 = \frac{w_0}{w_1 - z_0\sigma},
\]

\[
f_2 = \frac{w_2}{w_1 - z_1\sigma}, \quad g_2 = \frac{w_2}{w_0 - z_1\sigma}, \tag{30}
\]

where the value of \( z_0 \) is zero. Let us denote the numerators of equations (28) and (29) as \( N_1 \) and \( N_2 \) and the respective denominators \( D_1 \) and \( D_2 \). Following the same procedure, the value of \( N_1D_2 - N_2D_1 \) is obtained as

\[
N_1D_2 - N_2D_1 = \frac{C(0)K_1(0)w_2T_{cp}z_1\sigma(w_0 - w_1)}{(w_1 - z_1\sigma)(w_0 - z_1\sigma)}. \tag{31}
\]
From the above, we can see that if $w_1 \leq w_0$, then $(N_1D_2 - N_2D_1) \geq 0$, which implies that $\Gamma(T(p_0)) \geq \Gamma(T_c(p_1))$. Hence, if $w_1 \leq w_0$, by interchanging the processors $p_0$ and $p_1$, the processing time will decrease or remain the same if $w_1 < w_0$, which implies a definite decrease in the processing time.

**Theorem 4 (Optimal Arrangement).** Given a set of $(m + 1)$ processors, and $m$ links to be arranged in a single-level tree architecture following the nonblocking mode communication, then the processing time will be minimum if the processors and links are arranged in such a way that $w_0 \leq w_1$, $w_{i+1} \geq w_i$, and $z_{i+1} \geq z_i$ for $i = 1, 2, \ldots, m - 1$.

**Proof.** This theorem can be easily proved by a contradiction using Theorem 3 and Lemma 3.

## 4. RESULTS AND DISCUSSIONS

In all the earlier studies in scheduling divisible loads, the blocking mode of communication is used. In this paper, the nonblocking mode communication is presented for scheduling divisible loads. In this paper, a closed-form expression for the processing time of a divisible load in the nonblocking mode of communication is derived. Using this closed-form expression, important results on optimal sequencing and optimal arrangement are obtained. The processing time of a divisible load in the nonblocking mode of communication is always less than the processing time of the same divisible load in the blocking mode of communication. This is due to the fact that, in the the nonblocking mode of communication, the child processors will start the computation process, while the front-end is receiving the processing load. Because of this fact, the “idle time” of the child processors is reduced in the nonblocking mode of communication, in comparison with the blocking mode of communication. Idle time ($I_i$) of child processor $p_i$ is the time difference between the computation of time the processor $p_i$ and the computation of time the root processor $p_0$.

$$I_i = \alpha_0 w_0 T_{cp} - \alpha_i w_i T_{cp}, \quad i = 1, 2, \ldots, m. \quad (32)$$

Also, the utilization ($U_i$) of the child processor $p_i$ is increased in this nonblocking mode of communication compared to the blocking mode of communication. Utilization of a processor $p_i$ is defined as the ratio of the computation time of $p_i$ to the computation time of root processor $p_0$.

$$U_i = \frac{\alpha_i w_i T_{cp}}{\alpha_0 w_0 T_{cp}}, \quad i = 0, 1, \ldots, m. \quad (33)$$

**Numerical Example 1.** We now present a small numerical example. Consider a single-level tree network with four ($m = 4$) child processors. All the processors in the network are equipped with front-end processors. The computation speed parameters of the processors are $w_0 = 1.0$, $w_1 = 1.2$, $w_2 = 1.5$, $w_3 = 1.7$, and $w_4 = 1.9$. The communication speed parameters of the links are $z_1 = 0.2$, $z_2 = 0.3$, $z_3 = 0.4$, and $z_4 = 0.5$. And let $T_{cm} = T_{cp} = 1.0$. This network satisfies all the conditions on optimal sequencing and optimal arrangement discussed in the earlier section. The measures of interest processing time, idle time, and the utilization of processors are shown in Table 1(a) for the nonblocking mode of communication and in Table 1(b) for the blocking mode of communication. From this table, we can see that all the measures of interest in the nonblocking mode of communication are better than in the blocking mode of communication. Note that the values of $f_i$s are different in the blocking and nonblocking modes of communication.

**Numerical Example 2.** We now present another numerical example to show the effect of sequencing and arrangement of links and processors. Consider a single-level tree network with three child processors, $m = 3$. All the processors are equipped with front-end processors and let $T_{cp} = T_{cm} = 1$. The processor and link speed parameters for several single-level tree networks and the corresponding processing times are shown in Table 2. In this table, Case 1 shows the
Divisible Load Scheduling

Table 1. Performance measures of interest.

<table>
<thead>
<tr>
<th>Processor $p_j$</th>
<th>$p_0$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_j$</td>
<td></td>
<td>1.2</td>
<td>1.5</td>
<td>1.7</td>
<td>1.9</td>
</tr>
<tr>
<td>Load Fraction $a_j$</td>
<td>0.3279</td>
<td>0.2733</td>
<td>0.1822</td>
<td>0.1286</td>
<td>0.0880</td>
</tr>
<tr>
<td>Idle Time $I_j$</td>
<td>0</td>
<td></td>
<td>0.05465</td>
<td>0.10931</td>
<td>0.16075</td>
</tr>
<tr>
<td>Utilization $U_j$</td>
<td>1</td>
<td></td>
<td>0.8333</td>
<td>0.6666</td>
<td>0.5098</td>
</tr>
</tbody>
</table>

(b) Blocking mode of communication.

<table>
<thead>
<tr>
<th>Processor $p_j$</th>
<th>$p_0$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_j$</td>
<td></td>
<td>1.4</td>
<td>1.8</td>
<td>2.1</td>
<td>2.4</td>
</tr>
<tr>
<td>Load Fraction $a_j$</td>
<td>0.3608</td>
<td>0.2577</td>
<td>0.1718</td>
<td>0.1228</td>
<td>0.0869</td>
</tr>
<tr>
<td>Idle Time $I_j$</td>
<td>0</td>
<td>0.05154</td>
<td>0.10308</td>
<td>0.15218</td>
<td>0.19564</td>
</tr>
<tr>
<td>Utilization $U_j$</td>
<td>1</td>
<td>0.8571</td>
<td>0.7143</td>
<td>0.5782</td>
<td>0.4577</td>
</tr>
</tbody>
</table>

Table 2. Optimal sequence and arrangements.

<table>
<thead>
<tr>
<th>Case</th>
<th>$w_0$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$z_3$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1.7</td>
<td>1.0</td>
<td>1.2</td>
<td>1.5</td>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3905</td>
</tr>
<tr>
<td>II</td>
<td>1.7</td>
<td>1.2</td>
<td>1.0</td>
<td>1.5</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3782</td>
</tr>
<tr>
<td>III</td>
<td>1.7</td>
<td>1.0</td>
<td>1.2</td>
<td>1.5</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3766</td>
</tr>
<tr>
<td>IV</td>
<td>1.0</td>
<td>1.7</td>
<td>1.2</td>
<td>1.5</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3617</td>
</tr>
<tr>
<td>V</td>
<td>1.0</td>
<td>1.2</td>
<td>1.5</td>
<td>1.7</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3595</td>
</tr>
</tbody>
</table>

In earlier studies on divisible load scheduling [17,18], it is shown that it is possible to improve performance by eliminating some of the processor-link pairs from the network, which is the same as assigning zero loads to these processors. To identify such processor-link pairs, the concept of equivalent network is used in [17,18]. The set of processors $p_{i+1}, \ldots, p_{i+r}$ can be replaced by a single equivalent processor $p(i+1, \ldots, i+r)$ with equivalent processing speed parameters $w_{eq} = w(i+1, \ldots, i+r)$ and equivalent communication speed parameter $z_{eq} = z(i+1, \ldots, i+r)$. Using these $z_{eq}$ and $w_{eq}$ rules for optimal load distribution in the blocking mode of communication are presented in [17]. We can see from [17,18] that $z_{eq}$ and $w_{eq}$ derived for the blocking mode of communication are only functions of $f_i$. Hence, we can see that the rule for optimal load distribution given in [17] are valid for the nonblocking mode of communication also. But, note that the actual value of $f_i$ in the blocking mode is $(w_iT_{cp} + z_iT_{cm})/w_{i-1}T_{cp}$ and the value of $f_i$ in nonblocking mode is $w_iT_{cp}/(w_{i-1}T_{cp} - z_{i-1}T_{cm})$. Another condition is also needed in nonblocking mode $w_iT_{cp} > z_iT_{cm}$, which is discussed in this paper.

**DISCUSSIONS.** The details of implementing the nonblocking mode of communication in parallel processing systems are discussed in [25,26]. In our study, it does not imply one packet for one piece of data so as multiple packets for multiple pieces of data. Even though a piece of data may require multiple packets for send-receive-ack protocol if the data size is larger than allowable payload for
one packet. For example, the maximum payload of an ethernet is 1500 bytes, maximum ATM payload size 48 bytes. However, the size of one fingerprint image uncompressed is more than 1MB or around hundreds of KB when compressed. In addition, the total load to be processed is a large amount of images. Thus, sending fingerprint images frame-by-frame or piece-by-piece may incur additional protocol packets slightly but never considerably. In this context, the extra overhead due to piece-by-piece sending can be ignored. However, consideration of packets overheads of the send-receive-ack protocol is an interesting future subject of study.

Optimal sequence of load distribution discussed in this paper can be achieved without effecting any architectural rearrangement. Rearranging the links and the processors may not always be possible. If architectural rearrangement is permitted then it is possible to improve the time performance of the network further.

5. CONCLUSIONS

In this paper, scheduling divisible loads, in the nonblocking mode of communication, in a single-level tree network is considered. We have proved some important results concerning optimal load distribution, optimal sequencing of the processing load, and optimal arrangement of processors and links. It is shown that the performance measures of interest in this nonblocking mode of communication are better than in the blocking mode of communication.

REFERENCES


