UNIVERSITY AT STONY BROOK

CEAS Technical Report 808

Naive vesrus Optimal Scheduling for Data Intensive Applications

Kwangil Ko and Thomas G. Robertazzi

July 17, 2003

Naive versus Optimal Scheduling for Data Intensive Applications

Kwangil Ko and Thomas G. Robertazzi, Senior Member IEEE

Dept. of Electrical and Computer Engineering

Stony Brook University

Stony Brook, NY 11794

Phone: 631-632-8412/8400 Fax: 631-632-8494

E-mail: tom@ece.sunysb.edu

July 23, 2003

ABSTRACT

A new model for naive equal allocation of divisible computation and communication load is developed. The model includes a detailed accounting of solution reporting time. It is compared to sequential scheduling and a new type of multi-installment scheduling. Numerical results indicate that speedup improvements over equal division scheduling achievable through the use of optimal sequential scheduling can be at least as high as thirty percent and can be at least as

high as seventy percent for the type of multi-installment scheduling discussed here compared to equal division scheduling. Aerospace applications include the processing of satellite imagery, radar and sensor networks.

I. Introduction

The combination of the cost decrease and performance improvement in both computers and data storage devices has led to new data intensive applications in the aerospace field. Examples include processing satellite imagery, radar and sensor networks. It is becoming more common to conceive and implement systems processing on the order of a petabyte (i.e. 10^{15} bytes) of data a year.

In this paper we seek to examine a question for which answers to date have been anecdotal in nature. That is, to what extent can optimal scheduling improve performance in a parallel processor compared to naive scheduling? The naive scheduling polity investigated here is "equal division" scheduling where 1/N of the data load is assigned to each of N processors. The optimal policies discussed here are based on previously developed principles of divisible load scheduling.

In a divisible model, load is assumed to be completely partitionable (divisible) in terms of both computation and communication. Model parameters include processor and link speed(s), and computation and communication intensity. A specific model is also characterized by the parallel processor interconnection topology, scheduling policy and load distribution assumptions. Divisible load scheduling analysis makes use of linear and continuous variable mathematics to produce a tractable model. Typically one seeks to solve a particular model for the optimal allocation of load, speed-up and solution time. The study of divisible load models began in 1988 with a paper by Cheng and Robertazzi [8]. A 2003 survey paper on this topic is [3] and a 1996 monograph on the subject is [4].

The target architecture used in this paper is an L-level K-ary tree topology. Using a tree

topology is quite generic as any arbitrary interconnection topology can be spanned by a tree. Also if L=1 and all link speeds are equal, the tree reduces to a bus architecture.

The paper presents the first published closed form results for speedup for a multilevel tree network under equal division scheduling. This is compared with optimal single installment and multiple installment scheduling. We find improvements in speedup under optimal scheduling of as large as 70%.

Divisible load models involving single installment load distribution for trees were first considered in 1990 by Cheng and Robertazzi [7]. Load distribution sequencing in trees is discussed by Kim, Jee and Lee in [10]. The use of multiple installments of load distribution in tree networks was first examined by Ghose, Mani and Bharadwaj in 1995 [5]. Asymptotic results for large trees using the single installment policy by Bataineh and Robertazzi appeared in 1997 [1]. Asymptotic multi-installment results appear in [9]. Finally, the concept of an equivalent processor, used in this paper was introduced in Robertazzi [11]. A proof that optimal load allocation can be found by forcing all processors to stop computing at the same instant is presented in [12].

This paper is organized as follows. The system model is presented in section II. Equal division scheduling, sequential optimal scheduling and multi-installment scheduling are modeled in sections III, IV, and V, respectively. Numerical results appear in section VI. The conclusion is in section VII.

II. System Model of L-level K-ary Tree Network

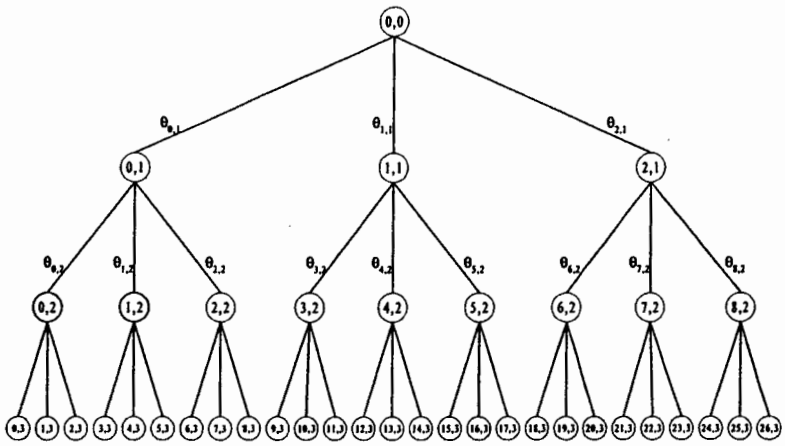

Figure 1: 3-level 3-ary tree network.

A L-level K-ary tree network of communicating processors is considered. For an example, a 3-level, 3-ary tree network is shown in Fig. 1. Each processor is labeled in terms of indexes from left to right and level to level. Here $p_{i,j}$ is the ith processor at the jth level. Processor 0 at level 0 is assumed to be the originating (root) processor which distributes the fractions of the entire load to K processors. All processors in the L^{th} level are terminal nodes and other processors each have K children processors. Non-terminal processors redistribute the fractions of the received load from the parent processor to K children processors. Also $\theta_{i,j}$ designates load distributed to processor i at level j (as discussed in section IV). It is assumed that communication speeds are high enough relative to computation speeds that eliminating subtrees does not result in a speedup improvement [3].

There exists K^j processors at the j^{th} level for $j = 0, 1, 2, \cdots, L$. Thus, the model of a L-level K-ary tree network consists of $\sum_{j=0}^{L} K^j$ processors. Without loss of generality, it will be

assumed that the load is instantaneously available at processor 0 at time 0. Each processor is interfaced with the network via a front-end communication processor for communication off-loading. That is, the processors can communicate and compute at the same time.

It is important for $p_{i,j}$ to know its parent processor since $p_{i,j}$ receives load fractions from its parent processor. Naturally, the parent processor of $p_{i,j}$ is located at the $(j-1)^{th}$ level just above the j^{th} level. The integer part of $\frac{i}{K}$ indicates the order (index) of parent processor as all processors at the $(j-1)^{th}$ level have K children processors. Thus, the parent processor of any processor, $p_{i,j}$ for $j = 1, 2, \cdots, L$ and $i = 0, 1, \cdots, K^j - 1$ is $p_{int(i/K),j-1}$. Let $P_P(p_{i,j})$ be the parent processor of $p_{i,j}$.

$$P_P(p_{i,j}) = p_{int(i/K),j-1} \tag{1}$$

Here, $int(\cdot)$ is the rounding down to the nearest integer. Further the grandparent processor of $p_{i,j}$ for any $j = 1, 2, \cdots, L$ and $i = 0, 1, \cdots, K^j - 1$ is $P_P[P_P(p_{i,j})]$.

$$\begin{aligned} P_P[P_P(p_{i,j})] &= P_P\left(p_{int(i/K),j-1}\right) \\ &= P_{int(int(i/K)/K),j-2} \\ &= P_{int(i/K^2),j-2} \end{aligned} \tag{2}$$

Generally, the ancestor processor of $p_{i,j}$ at the l^{th} level($l < j$) is $(j-l)$ levels above the j^{th} level. In a manner similar to equation (2), the ancestor processor of $p_{i,j}$ at the l^{th} level defined as $A_P^l(P_{i,j})$ is expressed as follows:

$$A_P^l(P_{i,j}) = p_{int(i/K^{j-l}),l} \tag{3}$$

This expression allows one to identify the ancestor processors of any processor, $p_{i,j}$ and perceive which processor distributes load fractions to it.

Alternately, a processor, $p_{i,j}$ for any $j = 0, 1, \cdots, L-1$ and $i = 0, 1, \cdots, K^j - 1$ has K children processors labeled $p_{iK+m,j+1}$ for $m = 0, 1, \cdots K-1$.

The following notation will be used throughout this paper:

- $p_{i,j}$: The i^{th} processor on the j^{th} level.
- $\alpha_{i,j}$: The fraction of the entire processing load that is assigned to the i^{th} processor on the j^{th} level.
- $w_{i,j}$: A constant inversely proportional to the computation speed of the i^{th} processor on the j^{th} level (see Fig 1).
- $z_{i,j}$: A constant inversely proportional to the channel speed of the i^{th} link at the j^{th} level.
- T_{cp}: Computing intensity constant. The entire load is processed in $w_{i,j}T_{cp}$ seconds by the i^{th} processor on the j^{th} level.
- T_{cm}: Communication intensity constant. The entire load can be transmitted in $z_{i,j}T_{cm}$ seconds over the i^{th} link at the j^{th} level.
- T_{cm}^{sol}: Solution reporting communication intensity constant. The entire solution report can be transmitted in $z_{i,j}T_{cm}^{sol}$ seconds over the i^{th} link at the j^{th} level.

III. Equal Division Scheduling

A multilevel tree is considered where load is distributed from the root to the children in a store and forward mode of operation and "solutions" are transmitted back to the root.

Each processor transmits load fractions to its children processors in sequence. That is, each processor transmits all the load that its left child (and its children) will require, then it does the same for the next (to the right) child and so on. Each processor, that is not a terminal node, repeats this load distribution policy. Thus, although load originates at the root, as load distribution proceeds multiple nodes in the tree will be concurrently distributing load. In equal division load scheduling, each processor keeps the same fraction of the total load for processing. A L-level K-ary tree network has $\sum_{j=0}^{L} K^j$ processors. Let ε be the fraction assigned to any processor. Consequently the fraction of normalized load for each processor is obtained from the inverse of the total number of processors.

$$\varepsilon = \frac{1}{\sum_{j=0}^{L} K^j} \tag{4}$$

The root processor at level 0 keeps ε, a fraction of the total processing load for itself to compute and divides and distributes the remaining load among to its children processors at the next level. The processors at this level perform the same operation with the load they receive. This process continues until the processors located at the terminal nodes of the tree are assigned their share of the processing load.

Our goal is to find an expression for the solution (finish) time for the system described under equal division scheduling. Towards this end the following subsection shows how to calculate a communication delay for each processor. Each processor starts to process its load fraction as soon as receives it.

A. Communication delay for processor, $p_{i,j}$ to receive its load fraction from root processor

Communication delay is divided into three parts; one is the time delay incurred by the parent processor, the second is the time delay incurred by the previous brother processors (which are children of the parent node), and the third is the time taken for $p_{i,j}$ to receive its load fraction and load fractions for descendant processors. The time at which the processor, $p_{i,j}$ finishes receiving its load fraction is defined as $C_d(p_{i,j})$. Assume that the parent processor distributes load fractions to its children processor starting from the left to the right.

$$C_d(p_{i,j}) = t_r(p_{i,j}) + t_i(p_{i,j}) + t_p(p_{i,j}) \tag{5}$$

Here, in a different order, $t_r(p_{i,j})$, $t_i(p_{i,j})$ and $t_p(p_{i,j})$ are the times taken to receive load fractions over the link to $p_{i,j}$, the time delay incurred by the prior brothers processors, and the time delay incurred by the parent processor of $p_{i,j}$, respectively.

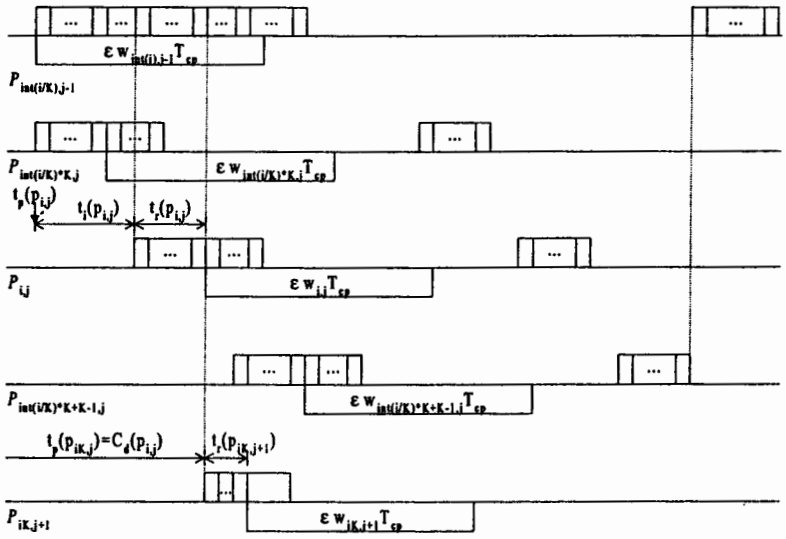

Figure 2: Timing diagram of equal division scheduling.

Fig. 2 illustrates equal division scheduling. In Fig. 2, the third row is for $p_{i,j}$. In the

diagram communication time appears above each axis and computation time appears below each axis. The first receiving brother processor is located on the second axis. The time taken to receive load fractions over the link to $p_{i,j}$ is shown on the third axis.

The difference between the instant that the first brother receiving processor begins to receive and the instant that $p_{i,j}$ begins to receive is $t_i\,(p_{i,j})$.

When $p_{i,j}$ finishes receiving load fractions, that time instant indicates the communication delay, $C_d\,(p_{i,j})$ for $p_{i,j}$. This is the same as $t_p\,(p_{iK+m,j+1})$, the time delay incurred by the processor $(p_{i,j})$ which is the parent of its children processors $(p_{iK+m,j+1})$ for $m = 0, 1, \cdots K-1$.

The root processor can process its load fraction while distributing the remaining load to its children processors. Thus there is no communication delay time for the root processor.

Next we develop expressions for the three components of $C_d\,(p_{i,j})$.

1. Receiving Time Delay, $t_r(p_{i,j})$

Each processor at the same level has the same number of children and grandchildren processors. Processors at any level except the L^{th} level have K children processors, K^2 grandchildren processors and so on. This is summed to the L^{th} level. The number of descendent processors for $p_{i,j}$ is defined as $N_D\,(p_{i,j})$.

$$N_D(p_{i,j}) = \sum_{m'=j+1}^{L} K^{m'-j} \tag{6}$$

$$= \sum_{m=1}^{L-j} K^{m} \tag{7}$$

$$= \frac{K - K^{L-j+1}}{1-K} \tag{8}$$

The processor, $p_{i,j}$ receives $[1 + N_D(p_{i,j})]$ fractions for itself and for its descendent proces-

sors from its parent processor and then sends $N_D(p_{i,j})$ fractions to its descendent processors. Because $p_{i,j}$ receives $[1 + N_D(p_{i,j})]$ fractions from its parent processor, $t_r(p_{i,j})$ is expressed as follows:

$$
\begin{aligned}
t_r(p_{i,j}) &= \varepsilon\left[1 + N_D(p_{i,j})\right] z_{i,j} T_{cm} && (9)\\
&= \varepsilon\left[1 + \sum_{m=1}^{L-j} K^m\right] z_{i,j} T_{cm} && (10)\\
&= \varepsilon \sum_{m=0}^{L-j} K^m \cdot z_{i,j} T_{cm} && (11)\\
&= \frac{1 - K^{L-j+1}}{1 - K} \varepsilon z_{i,j} T_{cm} && (12)
\end{aligned}
$$

Here, ε is the size of a fraction. The receiving time delay from the parent processor of $p_{i,j}$ depends on the level i, and only $z_{i,j}$, the inverse link speed connected to $p_{i,j}$.

2. Time delay of the prior brothers processors located at the same level, t_i

Now, if $p_{i,j}$ does not receive first on its level from its parent processor, $p_{i,j}$ should wait while its prior receiving brother processors at the same level receive load fractions from their parent processor. The remainder after dividing i by K decides the receiving order (position) of $p_{i,j}$. The processor, $p_{i,j}$ is the first receiving order processor at the j^{th} level when $mod\,(i/K)$ is zero. Here $mod\,(i/K)$ is the remainder after dividing i by K. Thus, the time delay due to the prior receiving processors can be expressed as follows:

$$
t_i\,(p_{i,j}) = \sum_{n=0}^{mod(i/K)-1} t_r(p_{inx(n,i),j}) \qquad (13)
$$

Here, $inx\,(n, i)$ is used to find the n^{th} receiving processor's index of brother processors of $p_{i,j}$. The indexes of processors at the j^{th} level are sequentially written starting from the first receiving child processor of $p_{0,j-1}$. From equation (1), the index of $P_P\,(p_{i,j})$, is $int\,(i/K)$.

Thus, the index of the children processors of $P_P(p_{i,j})$ starts from $int(i/K) \cdot K$. Furthermore, the n^{th} receiving processor index among brother processors of $p_{i,j}$ is obtained adding n and $int(i/K) \cdot K$. For $n = 0, 1, \cdots, mod(i/K) - 1$;

$$inx(n,i) = n + int(i/K) \cdot K \tag{14}$$

Applying equation (12) to (13), the time delay by the prior brother processors is obtained as follows:

$$t_i(p_{i,j}) = \varepsilon \sum_{n=0}^{mod(i/K)-1} \left[\sum_{m=0}^{L-j} K^m \cdot z_{inx(n,i),j} T_{cm} \right] \tag{15}$$

$$= \varepsilon \sum_{m=0}^{L-j} K^m \sum_{n=0}^{mod(i/K)-1} z_{inx(n,i),j} T_{cm} \tag{16}$$

$$= \varepsilon \frac{1 - K^{L-j+1}}{1 - K} \left[\sum_{n=0}^{mod(i/K)-1} z_{inx(n,i),j} T_{cm} \right] \tag{17}$$

The processors at the same level have the same number of load fractions as in equation (8) since they have the same number of descendant processors. The bracket in the above equation is the sum of the load distribution delays to the prior brother processors over their links.

3. Time delay by the parent processor of $p_{i,j}$, t_p

Each child processor has a time delay caused by waiting for its parent processor. This time delay, $t_p(p_{i,j})$ equals the total communication delay of the parent processor of $p_{i,j}$.

$$t_p(p_{i,j}) = C_d\left(A_P^{j-1}(p_{i,j})\right) \tag{18}$$

Here, $A_P^{j-1}(p_{i,j})$ is the parent processor of $p_{i,j}$. Using equation (5) and (18), equation, the following recursive equations can be obtained.

$$t_p(p_{i,j}) = C_d\left(A_P^{j-1}(p_{i,j})\right) \tag{19a}$$

$$C_d\left(A_P^{j-1}(p_{i,j})\right) = t_p\left(A_P^{j-1}(p_{i,j})\right) + t_i\left(A_P^{j-1}(p_{i,j})\right) + t_r\left(A_P^{j-1}(p_{i,j})\right) \quad (19b)$$

$$t_p\left(A_P^{j-1}(p_{i,j})\right) = C_d\left(A_P^{j-2}(p_{i,j})\right) \quad (19c)$$

$$C_d\left(A_P^{j-2}(p_{i,j})\right) = t_p\left(A_P^{j-2}(p_{i,j})\right) + t_i\left(A_P^{j-2}(p_{i,j})\right) + t_r\left(A_P^{j-2}(p_{i,j})\right) \quad (19d)$$

$$\vdots$$

$$t_p\left(A_P^{2}(p_{i,j})\right) = C_d\left(A_P^{1}(p_{i,j})\right) \quad (19e)$$

$$C_d\left(A_P^{1}(p_{i,j})\right) = t_p\left(A_P^{1}(p_{i,j})\right) + t_i\left(A_P^{1}(p_{i,j})\right) + t_r\left(A_P^{1}(p_{i,j})\right) \quad (19f)$$

$$t_p\left(A_P^{1}(p_{i,j})\right) = C_d\left(A_P^{0}(p_{i,j})\right) \quad (19g)$$

Summing both sides of the above recursive equations, $t_p(p_{i,j})$ can be rewritten as follows:

$$t_p(p_{i,j}) = C_d\left(A_P^0(p_{i,j})\right) + \sum_{l=1}^{j-1} t_r\left(A_P^l(p_{i,j})\right) + \sum_{l=1}^{j-1} t_i\left(A_P^l(p_{i,j})\right) \quad (20)$$

$$= \sum_{l=1}^{j-1} t_r\left(A_P^l(p_{i,j})\right) + \sum_{l=1}^{j-1} t_i\left(A_P^l(p_{i,j})\right) \quad (21)$$

Note that the root processor has no communication delay. Thus $C_d\left(A_P^0(p_{i,j})\right)$ is zero.

Substituting equation (21) into (5), communication delay for $p_{i,j}$, $C_d(p_{i,j})$, is expressed as follows:

$$C_d(p_{i,j}) = t_r(p_{i,j}) + t_i(p_{i,j}) \quad (22)$$

$$+ \sum_{l=1}^{j-1} t_r\left(A_P^l(p_{i,j})\right) + \sum_{l=1}^{j-1} t_i\left(A_P^l(p_{i,j})\right) \quad (23)$$

Note that $A_P^j(p_{i,j}) = p_{i,j}$.

$$C_d(p_{i,j}) = \sum_{l=1}^{j} \left[t_r\left(A_P^l(p_{i,j})\right) + t_i\left(A_P^l(p_{i,j})\right)\right] \quad (24)$$

Equation (3) is applied to (12) and (17):

$$t_r\left(A_P^l(p_{i,j})\right) = t_r\left(p_{int\left(i/K^{j-l}\right),l}\right)$$

$$= \varepsilon \frac{1-K^{L-l+1}}{1-K} z_{int\left(i/K^{j-l}\right),l} T_{cm} \quad (25)$$

and

$$t_i\left(A_P^l\left(p_{i,j}\right)\right) = t_i\left(p_{int\left(i/K^{j-l}\right),l}\right)$$
$$= \varepsilon\frac{1-K^{L-l+1}}{1-K}\sum_{n=0}^{mod\left[int\left(i/K^{j-l}\right)/K\right]-1} z_{inx\left[n,int\left(i/K^{j-l}\right)\right],l}T_{cm} \quad (26)$$

In the above equation, if $n = mod\left[int\left(i/K^{j-l}\right)/K\right]$, then $inx\left[n, int\left(i/K^{j-l}\right)\right] = int\left(i/K^{j-l}\right)$.

From equation (14):

$$inx\left\{mod\left[\frac{int\left(\frac{i}{K^{j-l}}\right)}{K}\right], int\left(\frac{i}{K^{j-l}}\right)\right\}$$
$$= mod\left[\frac{int\left(\frac{i}{K^{j-l}}\right)}{K}\right] + int\left(\frac{int\left(\frac{i}{K^{j-l}}\right)}{K}\right)\cdot K \quad (27)$$
$$= int\left(\frac{i}{K^{j-l}}\right) \quad (28)$$

Thus, the summation of $t_r\left(A_P^l\left(p_{i,j}\right)\right)$ and $t_i\left(A_P^l\left(p_{i,j}\right)\right)$ is mentioned.

$$t_r\left(A_P^l\left(p_{i,j}\right)\right) + t_i\left(A_P^l\left(p_{i,j}\right)\right)$$
$$= \varepsilon\frac{1-K^{L-l+1}}{1-K}\sum_{n=0}^{mod\left[int\left(i/K^{j-l}\right)/K\right]} z_{inx\left[n,int\left(i/K^{j-l}\right)\right],l}T_{cm} \quad (29)$$

Now equation (29) is substituted into (24).

$$C_d(p_{i,j}) = \sum_{l=1}^{j}\left[\varepsilon\frac{1-K^{L-l+1}}{1-K}\sum_{n=0}^{mod\left[int\left(i/K^{j-l}\right)/K\right]} z_{inx\left[n,int\left(i/K^{j-l}\right)\right],l}T_{cm}\right] \quad (30)$$

In the special case of homogeneous link speeds ($z_{i,j} = z$)

$$C_d(p_{i,j}) = \sum_{l=1}^{j}\left[\varepsilon\frac{1-K^{L-l+1}}{1-K}\sum_{n=0}^{mod\left[int\left(i/K^{j-l}\right)/K\right]} zT_{cm}\right] \quad (31)$$
$$= \sum_{l=1}^{j}\left[\varepsilon\frac{1-K^{L-l+1}}{1-K}\left[1+mod\left[\frac{int\left(\frac{i}{K^{j-l}}\right)}{K}\right]\right]\cdot zT_{cm}\right] \quad (32)$$

B. Closed Form of Finish Time

Here, since load is equally divided among processors and since a homogeneous network is considered, different processors may finish computing at different times.

The last receiving processor is $p_{K^L-1,L}$ in the L level K ary tree network. As soon as this node finishes processing its load fraction, the node reports its solution. The system is finished when the solution of $p_{K^L-1,L}$ is delivered to the originating processor.

First the time delay for $p_{K^L-1,L}$ to report solution is considered. It takes $\varepsilon \cdot z_{K^L-1,L} T_{cm}^{sol}$ to transmit the solution of $p_{K^L-1,L}$ from this node at level L to $p_{K^{L-1}-1,L-1}$, its parent processor at level $L-1$. The parent processor of $p_{K^L-1,L}$ collects the solutions of K children processors and transmits $(1+K)$ solutions including its own solution to the ancestor processor at level $L-1$. This procedure keeps until the originating processor receives all solutions. Let S_d be the time delay for $p_{K^L-1,L}$ to report its solution to the originating processor.

$$\begin{aligned}\frac{S_d}{\varepsilon T_{cm}^{sol}} &= 1 \cdot z_{K^L-1,L} + (1+K)\, z_{K^{L-1}-1,L-1} + \left(1+K+K^2\right) z_{K^{L-2}-1,L-2} + \cdots \\ &\quad + \left(1+K+K^2+\cdots+K^{L-1}\right) z_{K-1,1}\end{aligned}$$

The above equation can be condensed as follows:

$$S_d = \varepsilon \sum_{m=0}^{L-1} \sum_{n=0}^{m} K^n \cdot z_{K^{L-m}-1,L-m} T_{cm}^{sol} \tag{33}$$

$$= \varepsilon \sum_{m=0}^{L-1} \frac{1-K^{m+1}}{1-K} \cdot z_{K^{L-m}-1,L-m} T_{cm}^{sol} \tag{34}$$

For homogeneous network speeds:

$$S_d = \frac{\varepsilon z T_{cm}^{sol}}{1-K} \cdot \left[L - K\frac{1-K^L}{1-K}\right] \tag{35}$$

$$= \frac{\varepsilon z T_{cm}^{sol}}{K-1} \cdot \left[\frac{K^{L+1}-K}{K-1} - L\right] \tag{36}$$

The finish (solution) time for the equal division scheduling is obtained as follows:

$$T_f^{EDS}(L,K) = C_d(p_{L,K^L}) + \varepsilon w_{K^{L-1},L} T_{cp} + S_d \tag{37}$$

The first term is the communication delay for $p_{K^{L-1},L}$, the second term is the computation time for $p_{K^{L-1},L}$ and the third term is the reporting time.

IV. Sequential Optimal Scheduling

In sequential optimal scheduling, each processor that is not a terminal node distributes load to each child (once) in turn from left to right. The single transmission of load to a child includes all loads that child's descendents will need. Thus the "sequencing" is similar to equal division scheduling except the size of load fractions will now be determined optimally. In sequential optimal scheduling, solution reporting times are staggered in a subtree of a L-level K-ary tree network. Children processors finish reporting their solution while a parent processor is processing.

Load scheduling for the general tree network was first presented by Cheng and Robertazzi [7]. The general tree network can be collapsed into one equivalent processor that preserves the characteristics of the original tree network in terms of its minimum processing time. The concept of processor equivalence in [11] can be used to obtain a closed-form solution to the minimum processing time for tree networks (without solution time).

In this section, the closed form of the finish time for a L level K ary tree network is considered. The scheduling discussed here includes solution reporting time. The procedure to obtain the finish time for a L level K ary tree network can be expanded to a general tree network.

First, consider a subtree of the network that consists of one processor, $p_{i,L-1}$ for any $i = 0, 1, \cdots, K^{L-1} - 1$ at the $(L-1)^{th}$ level and K children processors which are terminal nodes of the network and labeled as $p_{iK+m,L}$ for $m = 0, 1, \cdots K-1$. This subtree can be analyzed as a single level tree network. It is assumed that a load of size $\theta_{i,L-1}$ is assigned to $p_{i,L-1}$ and its descendant processors. The size of $\theta_{i,L-1}$ is determined by the equivalent processor speed. The processor, $p_{i,L-1}$ receives $\theta_{i,L-1}$ from its parent processor. After that, this processor keeps a fraction $\alpha_{i,L-1}$ for itself to compute and distributes the remainder to its children processors. The first child receives a fraction $\alpha_{iK,L}$ of $\theta_{i,L-1}$, the second child receives a fraction $\alpha_{iK+1,L}$ of $\theta_{i,L-1}$, and so on until the last child receives a fraction $\alpha_{iK+K-1,L}$ of $\theta_{i,L-1}$.

The finish times for the parent processor, $p_{i,L-1}$ and the children processors $p_{iK+m,L}$ for $m = 0, 1, \cdots K-1$, considered in isolation, are given as follows:

$$T(\theta_{i,L-1}) = \theta_{i,L-1}\alpha_{i,L-1}w_{i,L-1}T_{cp} \tag{38}$$

This single level tree network can be collapsed into one equivalent processor that preserves the characteristics of the original tree network in terms of its minimum processing time. Define the $w_{i,j}^{eq}$ as a constant that is inversely proportional to the computation speed of the j^{th} level and i^{th} equivalent processor.

$$T(\theta_{i,L-1}) = \theta_{i,L-1}w_{i,L-1}^{eq}T_{cp} \tag{39}$$

From equation (38) and (39), the speed of the equivalent processor, $w_{i,L-1}^{eq}$ is obtained as follows for any $i = 0, 1, \cdots, K^j - 1$:

$$w_{i,L-1}^{eq} = \alpha_{i,L-1}w_{i,L-1} \tag{40}$$

Now, the inverse speed of the processor, $w_{i,L-1}$ is replaced by $w_{i,L-1}^{eq}$. Then, the processors at $(L-1)^{th}$ level are terminal processors. This same procedure is used for the subtrees at level $L-2$, $L-1$ and moves up to level 0, one level at a time. Every parent processor and its children will be collapsed into one equivalent processor. This process will continue until the root processor and its children are collapsed to one equivalent processor. Then the finish (solution) time of sequential optimal scheduling, our goal in this section, is obtained as follows:

$$T_f^{sos}(L,K) = w_{0,0}^{eq} T_{cp} \tag{41}$$

In the next section an expression is developed for the equivalent inverse processing speed, $w_{i,j}^{eq}$, of the ith equivalent node at the jth level under sequential optimal scheduling. This expression can be used to find $w_{0,0}^{eq}$, the inverse speed of a single equivalent processor for the entire multilevel tree network, and hence finish time and speedup. This is done by collapsing each single level subtree within the multilevel tree into an equivalent processor, starting at the bottom levels and working towards the root level. Finally a single equivalent processor of inverse speed $w_{0,0}^{eq}$ remains.

A. Equivalent Processor Speed

It is assumed that one parent processor and its children are collapsed into one equivalent processor from level L to level $j+1$. Thus, the processors at the $(j+1)$ level are terminal processors with equivalent processor speed, $w_{m,j+1}^{eq}$ for $m = 0, 1, \cdots, K^{j+1} - 1$. Fig. 3 shows a subtree with $p_{i,j}$ and $w_{iK+m,j+1}^{eq}$ for $m = 0, 1, \cdots, K-1$. The entire load assigned to $p_{i,j}$ and its descendant processors is $\theta_{i,j}$. Then, $p_{i,j}$ distributes the load fractions, $\theta_{iK+m,j+1}$ for $m = 0, 1, \cdots, K-1$ according to $w_{iK+m,j+1}^{eq}$.

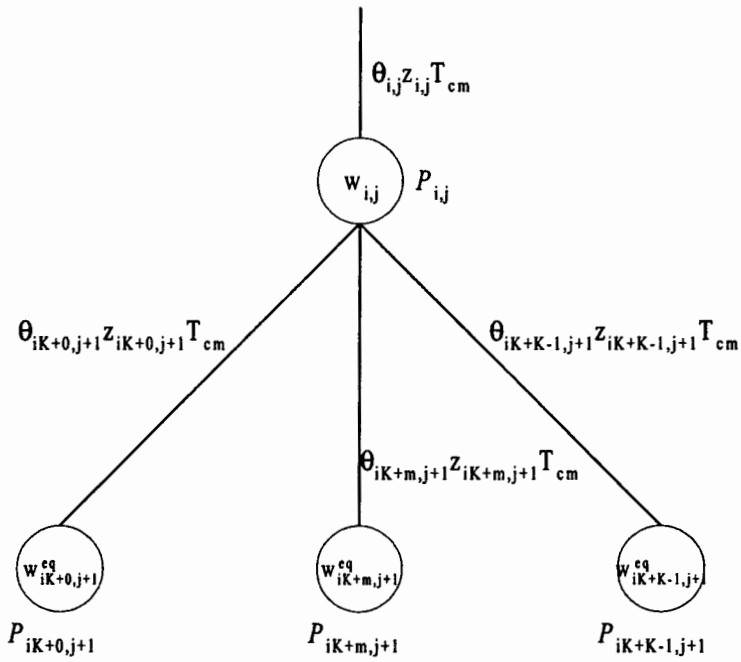

Figure 3: A subtree with equivalent processor.

It is assumed that solutions are reported in the same sequential as processors are received. The timing diagram for a subtree with equivalent processor is shown in Fig. 4.

From Fig. 4, the following recursive equation can be obtained for children processors of $p_{i,j}$, $p_{iK+m,j+1}$ for $m = 0, 1, \cdots, K-1$.

$$\theta_{iK+m,j+1} \cdot \left(w^{eq}_{iK+m,j+1} T_{cp} + z_{iK+m,j+1} T^{sol}_{cm} \right)$$

$$= \theta_{iK+m+1,j+1} \cdot \left(w^{eq}_{iK+m+1,j+1} T_{cp} + z_{iK+m+1,j+1} T_{cm} \right) \tag{42}$$

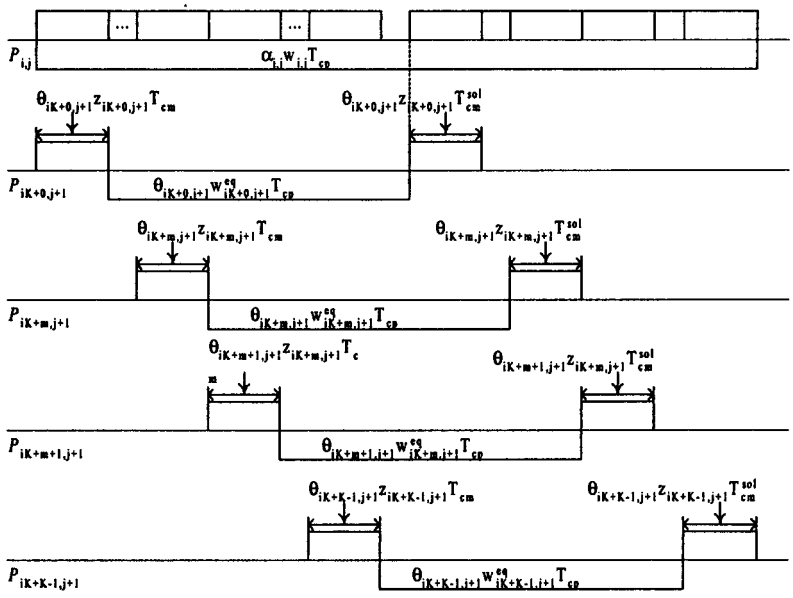

Figure 4: Timing Diagram of Sequential Optimal Scheduling for a Homogeneous Subtree Network.

or

$$\theta_{iK+m,j+1} = \theta_{iK+m+1,j+1} \frac{w^{eq}_{iK+m+1,j+1} T_{cp} + z_{iK+m+1,j+1} T_{cm}}{w^{eq}_{iK+m,j+1} T_{cp} + z_{iK+m,j+1} T^{sol}_{cm}} \tag{43}$$

The equation for the parent processor, $p_{i,j}$ is expressed as follows:

$$\begin{aligned} \alpha_{i,j} w_{i,j} T_{cp} &= \sum_{n=0}^{K-1} \theta_{iK+n,j+1} z_{iK+n,j+1} T_{cm} + \theta_{(i+1)K-1,j+1} w^{eq}_{(i+1)K-1,j+1} T_{cp} \\ &\quad + \theta_{(i+1)K-1,j+1} z_{(i+1)K-1,j+1} T^{sol}_{cm} \end{aligned} \tag{44}$$

Further, X_m is defined as follows:

$$X_m = \frac{w^{eq}_{iK+m+1,j+1} T_{cp} + z_{iK+m+1,j+1} T_{cm}}{w^{eq}_{iK+m,j+1} T_{cp} + z_{iK+m,j+1} T^{sol}_{cm}} \tag{45}$$

Now, $\theta_{iK+n,j+1}$ for $n = 0, 1, \cdots, K-1$ can be expressed in terms of $\theta_{(i+1)K-1,j+1}$. Equation

(43) can be rewritten as follows:

$$\theta_{iK+n,j+1} = \prod_{m=n}^{K-1} X_m \cdot \theta_{(i+1)K-1,j+1} \tag{46}$$

Applying equation (46) to (44), $\alpha_{i,j}$ equation (44) can be expressed in terms of $\theta_{(i+1)K-1,j+1}$.

$$\begin{aligned} \alpha_{i,j} &= \frac{\theta_{(i+1)K-1,j+1}}{w_{i,j}T_{cp}} \left[\sum_{n=0}^{K-1} \left(\prod_{m=n}^{K-1} X_m \right) z_{iK+n,j+1} T_{cm} \right. \\ &\quad \left. + w^{eq}_{(i+1)K-1,j+1} T_{cp} + z_{(i+1)K-1,j+1} T^{sol}_{cm} \right] \end{aligned} \tag{47}$$

The normalization equation is:

$$\sum_{n=0}^{K-1} \theta_{iK+n,j+1} + \alpha_{i,j} = \theta_{i,j} \tag{48}$$

Now $\theta_{(i+1)K-1,j+1}$ is obtained by substituting equation (46) and (47) into the above equation.

$$\begin{aligned} \theta_{(i+1)K-1,j+1} &= \theta_{i,j} \cdot \left[\sum_{n=0}^{K-1} \left(\prod_{m=n}^{K-1} X_m \right) + \sum_{n=0}^{K-1} \left(\prod_{m=n}^{K-1} X_m \right) \frac{z_{iK+n,j+1} T_{cm}}{w_{i,j} T_{cp}} \right. \\ &\quad \left. + \frac{w^{eq}_{(i+1)K-1,j+1} T_{cp}}{w_{i,j} T_{cp}} + \frac{z_{(i+1)K-1,j+1} T^{sol}_{cm}}{w_{i,j} T_{cp}} \right]^{-1} \end{aligned} \tag{49}$$

Equation (49) can be substituted into (47). Then $\alpha_{i,j}$ is obtained as follows:

$$\alpha_{i,j} = \frac{\theta_{i,j} \cdot \left(\sum_{n=0}^{K-1} \left(\prod_{m=n}^{K-1} X_m \right) \frac{z_{iK+n,j+1} T_{cm}}{w_{i,j} T_{cp}} + \frac{w^{eq}_{(i+1)K-1,j+1} T_{cp}}{w_{i,j} T_{cp}} + \frac{z_{(i+1)K-1,j+1} T^{sol}_{cm}}{w_{i,j} T_{cp}} \right)}{\sum_{n=0}^{K-1} \left(\prod_{m=n}^{K-1} X_m \right) \left(1 + \frac{z_{iK+n,j+1} T_{cm}}{w_{i,j} T_{cp}} \right) + \frac{w^{eq}_{(i+1)K-1,j+1} T_{cp}}{w_{i,j} T_{cp}} + \frac{z_{(i+1)K-1,j+1} T^{sol}_{cm}}{w_{i,j} T_{cp}}} \tag{50}$$

Now equation (45) is substituted into the above equation. Similarly in equation (40), the equivalent processor speed can be obtained as follows:

$$w^{eq}_{i,j} = \alpha_{i,j} \cdot w_{i,j} \tag{51}$$

$$= w_{i,j} A/B \tag{52}$$

$$A = \sum_{n=0}^{K-1} \left(\prod_{m=n}^{K-1} \frac{w^{eq}_{iK+m+1,j+1} T_{cp} + z_{iK+m+1,j+1} T_{cm}}{w^{eq}_{iK+m,j+1} T_{cp} + z_{iK+m,j+1} T^{sol}_{cm}} \right) \frac{z_{iK+n,j+1} T_{cm}}{w_{i,j} T_{cp}} \tag{53}$$

$$+\frac{w^{eq}_{(i+1)K-1,j+1}T_{cp}}{w_{i,j}T_{cp}} + \frac{z_{(i+1)K-1,j+1}T^{sol}_{cm}}{w_{i,j}T_{cp}}$$

$$B = \sum_{n=0}^{K-1}\left(\prod_{m=n}^{K-1}\frac{w^{eq}_{iK+m+1,j+1}T_{cp} + z_{iK+m+1,j+1}T_{cm}}{w^{eq}_{iK+m,j+1}T_{cp} + z_{iK+m,j+1}T^{sol}_{cm}}\right)\left(1+\frac{zT_{cm}}{w_{i,j}T_{cp}}\right) \tag{54}$$

$$+\frac{w^{eq}_{(i+1)K-1,j+1}T_{cp}}{w_{i,j}T_{cp}} + \frac{zT^{sol}_{cm}}{w_{i,j}T_{cp}}$$

In the above equation, $w^{eq}_{i,j}$ is expressed using its original processor speed, $w_{i,j}$ and the equivalent children processor speeds, $w^{eq}_{iK+m,j+1}$ for $m = 0, 1, \cdots, K-1$. The above expression of inverse equivalent processing speed for a single level subtree within a multilevel tree can be used to collapse subtrees into equivalent processors until a single equivalent processor for the entire network remains. From this finish time (equation (41)) can be found.

V. Multi-Installment Optimal Scheduling

In the sequential and equal division distribution of the previous sections, a child processor receives load fractions at the same time for itself and for processors at the next level. This causes the processors at each level to have long idle time. In this section, a processor at the j^{th} level doesn't distribute all load at once to each descendent processors but instead distributes load in turns (installments) to its descendent processors. The basic concept of multi-installment optimal scheduling was developed originally by Bharadwaj, Ghose and Mani [4,5] as a way to reduce solution time by modifying the load distribution policy. In this work partial load is delivered in several installments (rounds) to each processor to minimize idle time. A somewhat different approach is taken in this paper, distributing load in complete

integral units to each individual processor but in "installments" to the processors in the tree as a whole. That is, each node including the root distributes load to each of (only) K processors in turn during each set of installments. During each succeeding installment load is distributed in integral units for another K processors. The process repeats until all of the tree's processors have received load.

This scheduling strategy is best illustrated by way of example. Let, again, i be the children number and j be the level number for processor $p_{i,j}$. . In a 3-level 3-ary tree network, for instance, the root processor, $p_{0,0}$ distributes fractions to children processor in the sequence of $p_{0,1}$, $p_{1,1}$, $p_{2,1}$. As soon as each processor at the first level receives its load fraction, it begins to process. Again $p_{0,0}$ distributes load fractions to $p_{0,1}$, $p_{1,1}$, and $p_{2,1}$ in sequence. As $p_{0,1}$, $p_{1,1}$, and $p_{2,1}$ already received their load fractions, these processors can redistribute load to their children processors. That is as soon as they receive fractions, $p_{0,1}$, $p_{1,1}$, and $p_{2,1}$ distribute load fractions to $p_{0,2}$, $p_{3,2}$, and $p_{6,2}$ respectively. After that, additional load fractions are distributed to $p_{0,1}$, $p_{1,1}$, and $p_{2,1}$. As $p_{0,2}$, $p_{3,2}$, and $p_{6,2}$ already received their load fractions, this time, $p_{0,1}$, $p_{1,1}$, and $p_{2,1}$ distribute load fractions to $p_{1,2}$, $p_{4,2}$, and $p_{7,2}$ respectively.

As $p_{0,0}$ distributes fractions in the sequence of $p_{0,1}$, $p_{1,1}$, $p_{2,1}$, processors $p_{0,2}$, $p_{3,2}$, and $p_{6,2}$ receive before $p_{1,2}$, $p_{4,2}$, and $p_{7,2}$ receive. The receiving order at the second level is $p_{0,2}$, $p_{3,2}$, $p_{6,2}$, then $p_{1,2}$, $p_{4,2}$, $p_{7,2}$, then $p_{2,2}$, $p_{5,2}$, and $p_{8,2}$. After each processor at the second level receives its load fraction, it begins to distribute the load fractions received from its parent processor. This procedure continues until the terminal processors receive their fraction.

This strategy shuffles the index, i in $p_{i,j}$. In Fig. 5, the number beside the link indicates

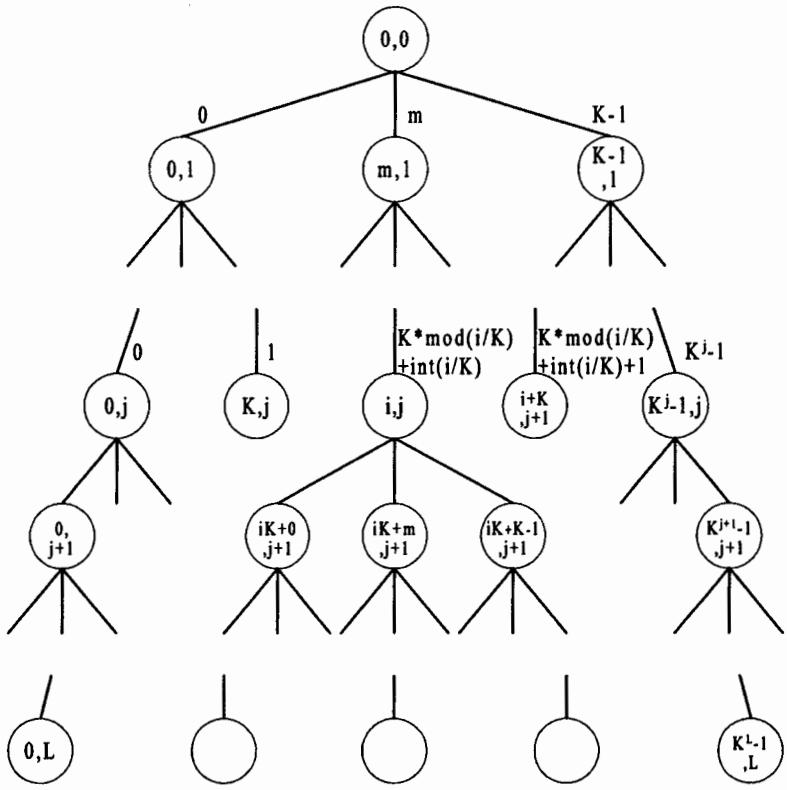

Figure 5: A L-level K-ary tree network.

the distribution sequence at the same level. Now, the actual distribution sequence at the j^{th} level can be calculated with the processor identification number, the index i in $p_{i,j}$.

$$p_{i,j} = p'_{K \cdot mod\left(\frac{i}{K}\right)+int\left(\frac{i}{K}\right),j} \tag{55}$$

or

$$p'_{m,j} = p_{K \cdot mod\left(\frac{m}{K}\right)+int\left(\frac{m}{K}\right),j} \tag{56}$$

Thus $p_{i,j}$ is the $\left(K \cdot mod\left(\frac{i}{K}\right) + int\left(\frac{i}{K}\right)\right)^{th}$ receiving processor at the j^{th} level. Let $p'_{m,j}$ be the m^{th} receiving processor at the j^{th} level. Furthermore, $\alpha'_{m,j}$, $w'_{m,j}$, and $z'_{m,j}$ are relative to $p'_{m,j}$. The prime variable is written in terms of the actual sequence of load distribution to

account for the load distribution shuffling of processor identification.

The goal in the following is to find an expression for the finish time, and hence speedup, of optimal multi-installment load distribution for the described multilevel tree network.

A. Load Distribution

1. Processors at the j^{th} Level

Fig. 6 shows the timing diagram for processors at the j^{th} level. While $p_{0,j}$, the first receiving processor is receiving its fraction, the second receiving processor, $p_{K,j}$, is idle until its grandparent finishes sending a load fraction for itself to its parent processor. As soon as $p_{0,j}$ finishes receiving its load fraction, the parent processor of $p_{K,j}$ distribute a fraction to $p_{K,j}$.

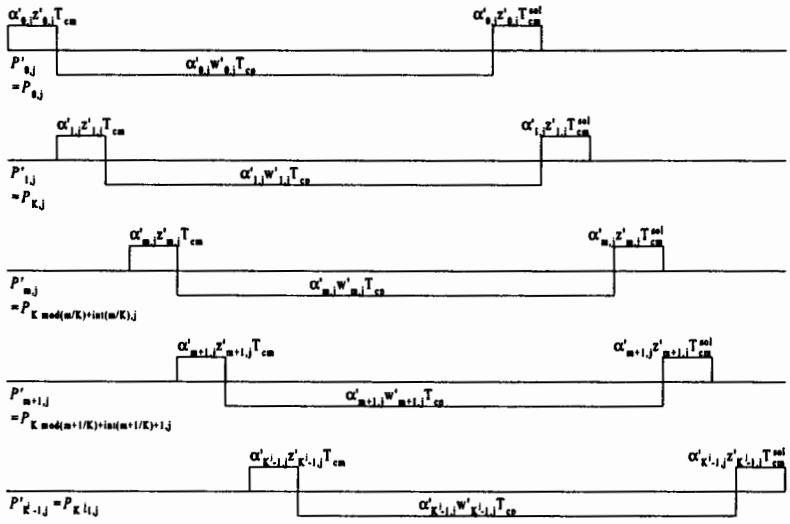

Figure 6: Timing diagram of multi-installment optimal scheduling for the j^{th} level of a L-level K-ary tree network.

For the processors $p'_{m,j}$ for $0 \leq m \leq K^j - 1$ at the j^{th} level it is assumed that solutions

are reported in the same sequential processor order that load is received in (without loss of generality they could be reported in the reverse order). From Fig. 6 the following recursive equations are obtained. For $1 \leq m \leq K^j - 1$:

$$\alpha'_{m,j} \cdot \left(w'_{m,j} T_{cp} + z'_{m,j} T^{sol}_{cm}\right) = \alpha'_{m+1,j} \cdot \left(w'_{m+1,j} T_{cp} + z'_{m+1,j} T_{cm}\right) \tag{57}$$

or:

$$\alpha'_{m,j} = \alpha'_{m+1,j} \frac{w'_{m+1,j} T_{cp} + z'_{m+1,j} T_{cm}}{w'_{m,j} T_{cp} + z'_{m,j} T^{sol}_{cm}} \tag{58}$$

$$= \alpha'_{m+1,j} \cdot X_{m,j} \tag{59}$$

Here:

$$X_{m,j} = \frac{\alpha'_{m,j}}{\alpha'_{m+1,j}} = \frac{w'_{m+1,j} T_{cp} + z'_{m+1,j} T_{cm}}{w'_{m,j} T_{cp} + z'_{m,j} T^{sol}_{cm}} \tag{60}$$

Thus, equation (57) can be rewritten in terms of $X_{m,j}$ for $0 \leq m \leq K^j - 2$.

$$\alpha'_{0,j} = \alpha'_{1,j} \cdot X_{0,j} \tag{61}$$

$$\vdots$$

$$\alpha'_{m,j} = \alpha'_{m+1,j} \cdot X_{m,j} \tag{62}$$

$$\vdots$$

$$\alpha'_{K^j-2,j} = \alpha'_{K^j-1,j} \cdot X_{K^j-1,j} \tag{63}$$

Now $\alpha'_{n,j}$ for $0 \leq n \leq K^j - 1$ can be rewritten in terms of $\alpha'_{0,j}$.

$$\alpha'_{n,j} = \prod_{m=0}^{n-1} X^{-1}_{m,j} \cdot \alpha'_{0,j} \tag{64}$$

Note that $\prod_{m=b}^{a} X^{-1}_{m,j}$ is one when $a < b$.

2. First Reception Processors at a level

Fig. 7 shows the timing diagram for the first reception processors at a level. After the fraction for $p_{Kj-1,j}$ at the j^{th} level finishes being distributed, the parent processor of $p_{0,j}$ intends to distributes another load fraction to $p_{0,j}$. The processor, $p_{0,j}$ sends the load fraction to $p_{0,j+1}$ as soon as it receives the fraction from its parent processor. However, there is a time gap between the distribution time of the last receiving processor at the j^{th} level and the distribution time of the first receiving processor at the $(j+1)^{th}$ level. The time gap is $\alpha_{0,j+1}z_{0,j}T_{cm}$ (see the fourth and sixth axis in Fig. 7). Note that $p'_{0,j} = p_{0,j}$ and $p'_{Kj-1,j} = p_{Kj-1,j}$ from equation (56)

From the fourth axis in Fig. 7 the processing time of the processor which receives its load fraction first at the j^{th} level is expressed as follows. For $j = 1, \cdots, L-1$:

$$\begin{aligned}\alpha'_{0,j}w'_{0,j}T_{cp} &= \sum_{n=1}^{Kj-1} \alpha'_{n,j}z'_{n,j}T_{cm} + \alpha'_{0,j+1}z'_{0,j}T_{cm} \\ &+ \sum_{n=0}^{Kj+1-1} \alpha'_{n,j+1}z'_{n,j+1}T_{cm} + \alpha'_{Kj+1-1,j+1}w'_{Kj+1-1,j+1}T_{cp} \\ &+\alpha'_{Kj+1-1,j+1}\left(z'_{Kj+1-1,j+1}T^{sol}_{cm} + z'_{Kj-1,j}T^{sol}_{cm}\right) \end{aligned} \quad (65)$$

The first term is the reception delay for the processors at the j^{th} level, the second term is the time gap mentioned above, the third term is reception delay for the processors at the $(j+1)^{th}$ level, the fourth term is computation time of the last receiving processor at the $(j+1)^{th}$ level, the last term is the solution reporting time of $p'_{Kj+1-1,j+1}$ over two links above the the $(j+1)^{th}$ and the j^{th} level. The time gap is occurred in reporting time. From equation (64), the following can be obtained.

$$\alpha'_{n,j} = \prod_{m=0}^{n-1} X^{-1}_{m,j} \cdot \alpha'_{0,j} \quad (66)$$

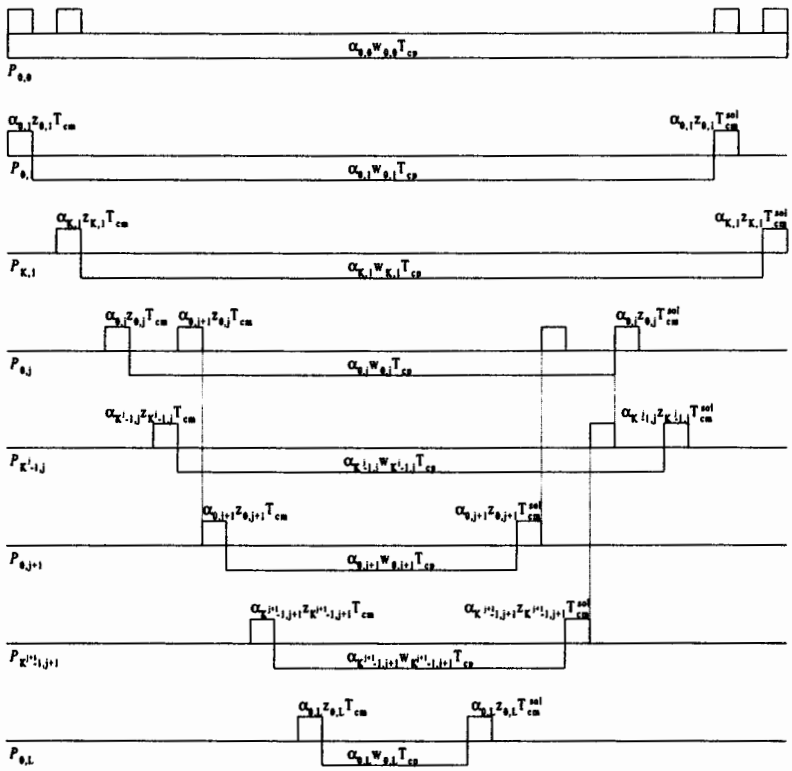

Figure 7: Timing diagram of multi-installment optimal scheduling for a L -level K-ary tree network.

$$\alpha'_{n,j+1} = \prod_{m=0}^{n-1} X_{m,j+1}^{-1} \cdot \alpha'_{0,j+1} \tag{67}$$

$$\alpha'_{K^{j+1}-1,j+1} = \prod_{m=0}^{K^{j+1}-2} X_{m,j+1}^{-1} \cdot \alpha'_{0,j+1} \tag{68}$$

The above equations are substituted into equation (65).

$$\begin{aligned} & \alpha'_{0,j} w'_{0,j} T_{cp} \\ = & \sum_{n=1}^{K^j-1} \left(\prod_{m=0}^{n-1} X_{m,j}^{-1} \cdot \alpha'_{0,j} \right) z'_{n,j} T_{cm} + \alpha'_{0,j+1} z'_{0,j} T_{cm} \\ & + \sum_{n=0}^{K^{j+1}-1} \left(\prod_{m=0}^{n-1} X_{m,j+1}^{-1} \cdot \alpha'_{0,j+1} \right) z'_{n,j+1} T_{cm} \end{aligned} \tag{69}$$

$$+\prod_{m=0}^{K^{j+1}-2} X_{m,j+1}^{-1} \cdot \alpha'_{0,j+1} \left(w'_{K^{j+1}-1,j+1} T_{cp} + z'_{K^{j+1}-1,j+1} T_{cm}^{sol} + z'_{K^j-1,j} T_{cm}^{sol} \right) \quad (70)$$

Equation (70) can be rewritten as follows:

$$\begin{aligned} & \alpha'_{0,j} \left[w'_{0,j} T_{cp} - \sum_{n=1}^{K^j-1} \left(\prod_{m=0}^{n-1} X_{m,j}^{-1} \right) z'_{n,j} T_{cm} \right] \\ = \; & \alpha'_{0,j+1} \left[z'_{0,j} T_{cm} + \sum_{n=0}^{K^{j+1}-1} \left(\prod_{m=0}^{n-1} X_{m,j+1}^{-1} \right) z'_{n,j+1} T_{cm} \right. \\ & \left. + \prod_{m=0}^{K^{j+1}-2} X_{m,j+1}^{-1} \cdot \left(w'_{K^{j+1}-1,j+1} T_{cp} + z'_{K^{j+1}-1,j+1} T_{cm}^{sol} + z'_{K^j-1,j} T_{cm}^{sol} \right) \right] \quad (71) \end{aligned}$$

or:

$$\alpha'_{0,j} = \alpha'_{0,j+1} Y_j \quad (72)$$

Here:

$$\begin{aligned} Y_j &= \frac{\alpha'_{0,j}}{\alpha'_{0,j+1}} \quad (73) \\ &= \left[z'_{0,j} T_{cm} + \sum_{n=0}^{K^{j+1}-1} \left(\prod_{m=0}^{n-1} X_{m,j+1}^{-1} \right) z'_{n,j+1} T_{cm} \right. \\ &\quad \left. + \prod_{m=0}^{K^{j+1}-2} X_{m,j+1}^{-1} \cdot \left(w'_{K^{j+1}-1,j+1} T_{cp} + z'_{K^{j+1}-1,j+1} T_{cm}^{sol} + z'_{K^j-1,j} T_{cm}^{sol} \right) \right] \quad (74) \\ &\div \left[w'_{0,j} T_{cp} - \sum_{n=1}^{K^j-1} \left(\prod_{m=0}^{n-1} X_{m,j}^{-1} \right) z'_{n,j} T_{cm} \right] \quad (75) \end{aligned}$$

Now the processing time of the root processor, and hence system finish time, can be found as follows (see the first axis in Fig.7):

$$\begin{aligned} \alpha'_{0,0} w_{0,0} T_{cp} &= \sum_{n=0}^{K-1} \alpha'_{n,1} z'_{n,1} T_{cm} \\ &\quad + \alpha'_{K-1,1} \cdot \left(w'_{K-1,1} T_{cp} + z'_{K-1,1} T_{cm}^{sol} \right) \quad (76) \end{aligned}$$

Here from equation (64):

$$\alpha'_{n,1} = \prod_{m=0}^{n-1} X_{m,1}^{-1} \cdot \alpha'_{0,1} \quad (77)$$

$$\alpha'_{K-1,1} = \prod_{m=0}^{K-2} X_{m,1}^{-1} \cdot \alpha'_{0,1} \tag{78}$$

Similarly, using equation (64), equation (76) can be rewritten as follows:

$$\alpha_{0,0} = Y_0 \alpha'_{0,1} \tag{79}$$

Here:

$$Y_0 = \frac{\sum_{n=0}^{K-1} \left(\prod_{m=0}^{n-1} X_{m,1}^{-1}\right) z'_{n,1} T_{cm} + \left(\prod_{m=0}^{K-2} X_{m,1}^{-1}\right) \left(w'_{K-1,1} T_{cp} + z'_{K-1,1} T_{cm}^{sol}\right)}{w_{0,0} T_{cp}} \tag{80}$$

Now $\alpha'_{0,j}$ for $j = 0, 1, \cdots, L-1$ can be expressed in terms of Y_j and $\alpha'_{0,j+1}$.

$$\alpha'_{0,0} = Y_0 \alpha'_{0,1} \tag{81}$$

$$\vdots$$

$$\alpha'_{0,j-1} = Y_{j-1} \alpha'_{0,j} \tag{82}$$

$$\alpha'_{0,j} = Y_j \alpha'_{0,j+1} \tag{83}$$

$$\vdots$$

$$\alpha'_{0,L-1} = Y_{L-1} \alpha'_{0,L} \tag{84}$$

From above equations, $\alpha'_{0,l}$ for $l = 1, 2, \cdots, L-1$ can be expressed in terms of $\alpha_{0,0}$.

$$\alpha'_{0,l} = \prod_{j=0}^{l-1} Y_j^{-1} \cdot \alpha_{0,0} \tag{85}$$

Furthermore $\alpha'_{n,l}$ can be express in terms of $\alpha'_{0,l}$ substituting (85) into (64).

$$\alpha'_{n,l} = \left(\prod_{m=0}^{n-1} X_{m,l}^{-1}\right) \cdot \left(\prod_{j=0}^{l-1} Y_j^{-1}\right) \cdot \alpha_{0,0} \tag{86}$$

The normalization equation is:

$$\alpha'_{0,0} + \sum_{l=1}^{L} \sum_{n=0}^{K'-1} \alpha'_{n,l} = 1 \tag{87}$$

Equation (86) is substituted into the above equation. Then α_0 can be obtained as follows:

$$\alpha'_{0,0} = \left[1 + \sum_{l=1}^{L} \sum_{n=0}^{K^l-1} \left(\prod_{m=0}^{n-1} X_{m,l}^{-1}\right) \cdot \left(\prod_{j=0}^{l-1} Y_j^{-1}\right)\right]^{-1} \tag{88}$$

The finish (solution) time then is:

$$T_f^{MOS}(L,K) = \alpha'_{0,0} w T_{cp} \tag{89}$$

$$= \frac{w T_{cp}}{1 + \sum_{l=1}^{L} \sum_{n=0}^{K^l-1} \left(\prod_{m=0}^{n-1} X_{m,l}^{-1}\right) \cdot \left(\prod_{j=0}^{l-1} Y_j^{-1}\right)} \tag{90}$$

For a homogeneous network speedup can be calculated as wT_{cp} divided by the above solution time for the complete multilevel tree.

VI. Numerical Results

Speedup, for a computational problem, is the ratio of solution time on one processor to solution time on N processors. It is thus a measure of parallel processing advantage.

The speedup versus the K and L for equal division scheduling, sequential optimal scheduling and multi-installment optimal scheduling are plotted in Fig. 8, 9, 10 and 11, respectively. The speedup increases as either K or L is increased.

Sequential scheduling and the multi-installment scheduling are compared with equal division scheduling. The speedup improvement is shown in Fig. 12 and 13. The speedup improvement is obtained as follows. For Fig. 12:

$$I_S|_{\text{Fig. 12}} = \frac{S^{SOS}(L,K) - S^{EDS}(L,K)}{S^{EDS}(L,K)} \times 100\,[\%] \tag{91}$$

For Fig. 13:

$$I_S|_{\text{Fig. 13}} = \frac{S^{MOS}(L,K) - S^{EDS}(L,K)}{S^{EDS}(L,K)} \times 100\,[\%] \tag{92}$$

Here, $S^{EDS}(L,K)$, $S^{SOS}(L,K)$ and $S^{MOS}(L,K)$ are the speedups for equal division scheduling, sequential scheduling and multi-installment scheduling, respectively. The speedup is defined as follow:

$$S^{EDS}(L,K) = \frac{wT_{cp}}{T_f^{EDS}(L,K)} \tag{93}$$

$$S^{SOS}(L,K) = \frac{wT_{cp}}{T_f^{SOS}(L,K)} \tag{94}$$

$$S^{MOS}(L,K) = \frac{wT_{cp}}{T_f^{MOS}(L,K)} \tag{95}$$

Also the speedup improvement calculated in equation (91) and (92) are shown in Table. 1 and 2. Five or six digits accuracy is shown here, not because real scheduling is that precise, but to aid in result replication. Comparing the multi-installment scheduling with the sequential scheduling, the multi-installment strategy has the higher speedup.

In the tables it can be seen that speedup improvements of from 3% to 70% were found for the tree topology. As L and K are increased the speedup improvement first increases then may decrease for certain parameter combinations.

It is interesting to ask over what range of parameter values is the speedup improvement most pronounced. If a job is computation intensive one would expect 1/N equal division scheduling to be optimal. If a job is communication intensive, the use of a single processor may well be optimal. It is that range where computation intensity is one the order of communication intensity that one can expect optimal scheduling to be most efficacious.

VII. Conclusion

The results in this paper indicate that one can achieve significant improvements in speedup (and by implication, solution time) by using optimal scheduling policies compared to naive

	$K=1$	$K=2$	$K=3$	$K=4$
$L=1$	3.0000	4.8657	6.6068	8.2321
$L=2$	7.7911	13.2612	18.8595	23.6171
$L=3$	14.0182	22.4653	29.0953	31.5415
$L=4$	21.3361	29.3861	32.7270	26.7681

Table 1: Speedup Improvement in Percentage: Equal Division Scheduling vs. Sequential Optimal Scheduling; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$

	$K=1$	$K=2$	$K=3$	$K=4$
$L=1$	3.0000	4.8657	6.6068	8.2321
$L=2$	9.5714	18.6093	26.8686	32.4850
$L=3$	20.2847	43.5657	49.8413	36.7902
$L=4$	34.6968	70.2669	42.7284	20.8299

Table 2: Speedup Improvement in percentage: Equal Division Scheduling vs. Multi-Installment Optimal Scheduling; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$

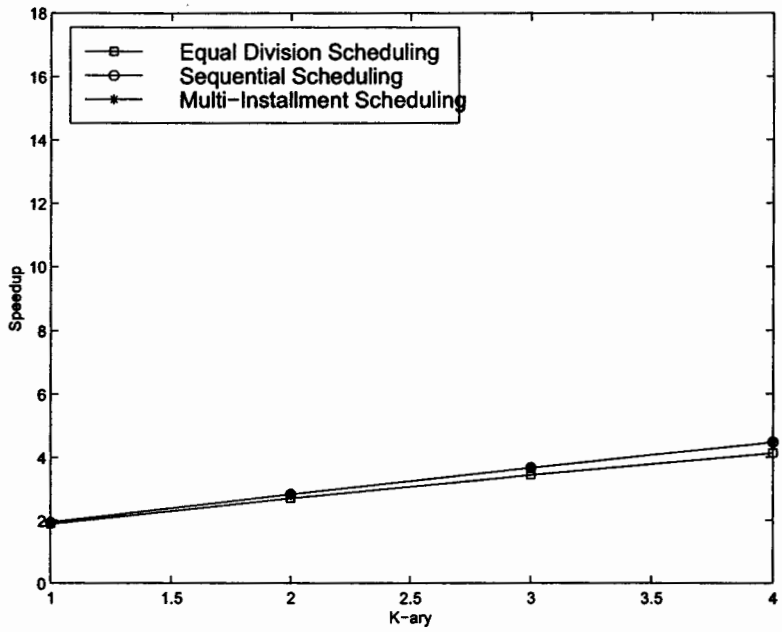

Figure 8: Speedup vs. K and L; $L = 1$; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$.

equal division scheduling. In doing this the first published analytical model for equal division scheduling for a multilevel tree network is presented. We believe the improvement noted will carry over to other topologies through the numerical amount of improvement will, of course, differ. Useful future work would include determining performance improvement bounds, both across possible parameter values for a particular topology and across different topologies.

VIII. Acknowledgements

The support of the National Science Foundation through grant CCR-99-12331 is acknowledged.

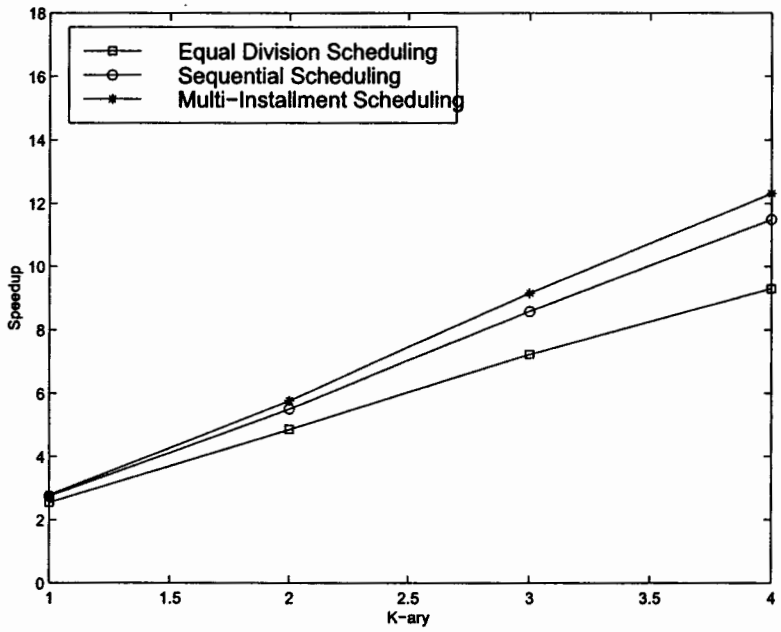

Figure 9: Speedup vs. K and L; $L = 2$; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$.

References

[1] Bataineh, S. and Robertazzi, T.G., (1997) Performance limits for processor networks with divisible jobs. *IEEE Transactions on Aeropsace and Electronic Systems*, **33** (1997), 1189-1198.

[2] Barlas, G.D. (1998) Collection-aware optimum sequencing of operations and closed-from solutions for the distribution of divisible load on arbitrary processor trees. *IEEE Transactions on Parallel and Distributed Systems*, **9** (1998), 929-941.

[3] Bharadwaj, V., Ghose, D. and Robertazzi, T.G. (2003) Divisible load theory: a new paradigm for load scheduling in distributed systems. *Cluster Computing*, **6** (2003).

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

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

[6] 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*, **5** (1994) 968-976.

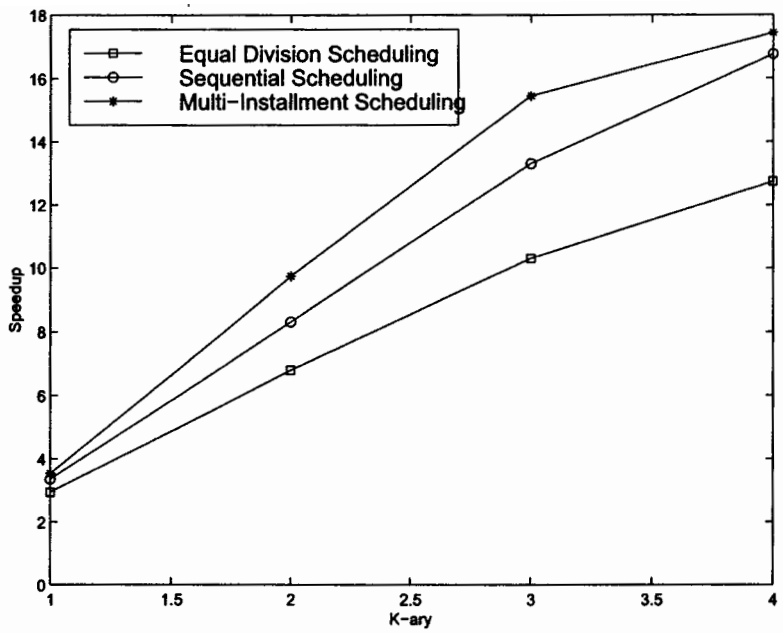

Figure 10: Speedup vs. K and L; $L = 3$; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$.

[7] Cheng, Y.-C. and Robertazzi, T.G. (1990) Distributed computation for a tree network with communication delays. *IEEE Transactions on Aerospace and Electronic Systems*, **26** (1990) 511-516.

[8] Cheng, Y.-C. and Robertazzi, T.G. (1988) Distributed computation with communication delays. *IEEE Transactions on Aerospace and Electronic Systems*, **24** (1988) 700-712.

[9] Ghose, D. and Mani, V. (1994) Distributed computation with communication delays: asymptotic performance analysis. *Journal of Parallel and Distributed Computing*, **23** (1994) 293-305.

[10] Kim, H.J., Jee, G.-I. and Lee, J.G. (1996) Optimal load distribution for tree network processors. *IEEE Transactions on Aerospace and Electronic Systems*, **32** (1996) 607-612.

[11] Robertazzi, T.G. (1993) Processor equivalence for a linear daisy chain of load sharing processors. *IEEE Transactions on Aerospace and Electronic Systems*, **29** (1993) 1216-1221.

[12] Sohn, J. and Robertazzi, T.G. (1996) Optimal load sharing for a divisible job on a bus network. *IEEE Transactions on Aerospace and Electronic Systems*, **32** (1996) 34-40.

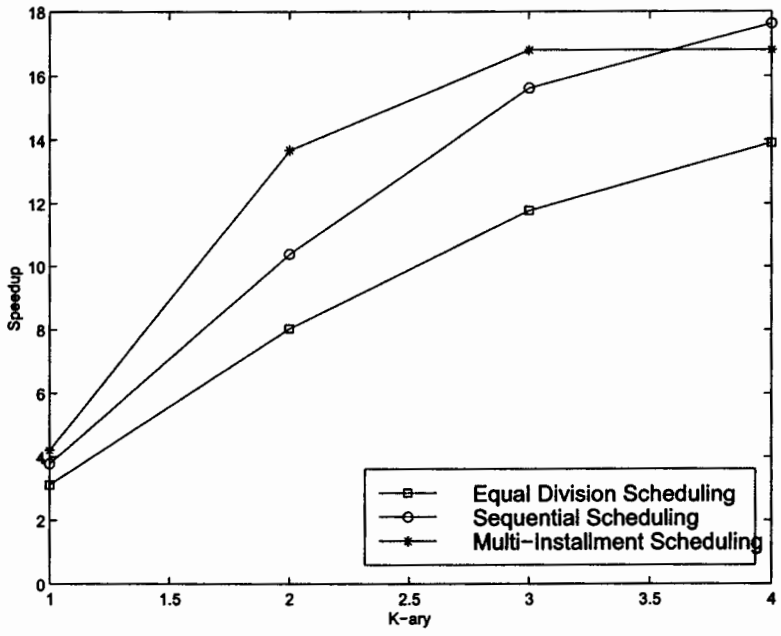

Figure 11: Speedup vs. K and L; $L = 4$; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$.

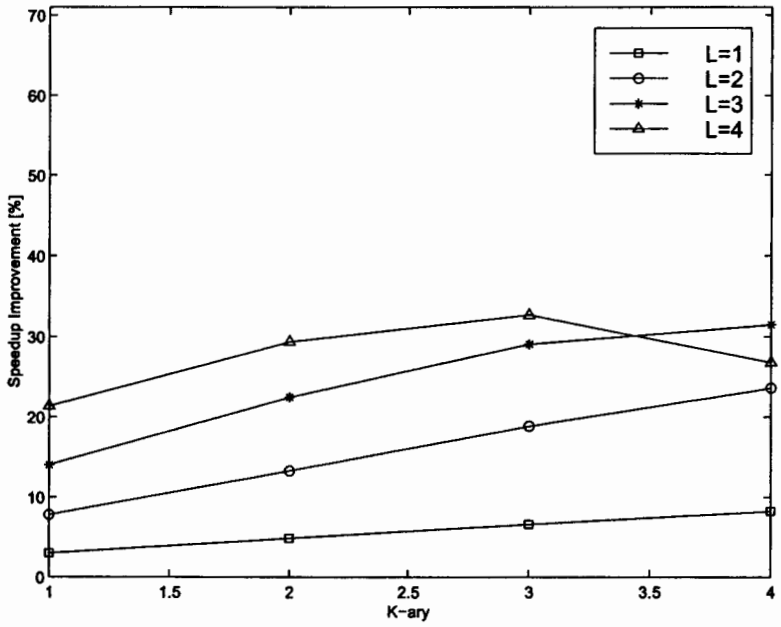

Figure 12: Speedup improvement in percentage vs. K and L; comparing equal division scheduling with sequential optimal scheduling; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$.

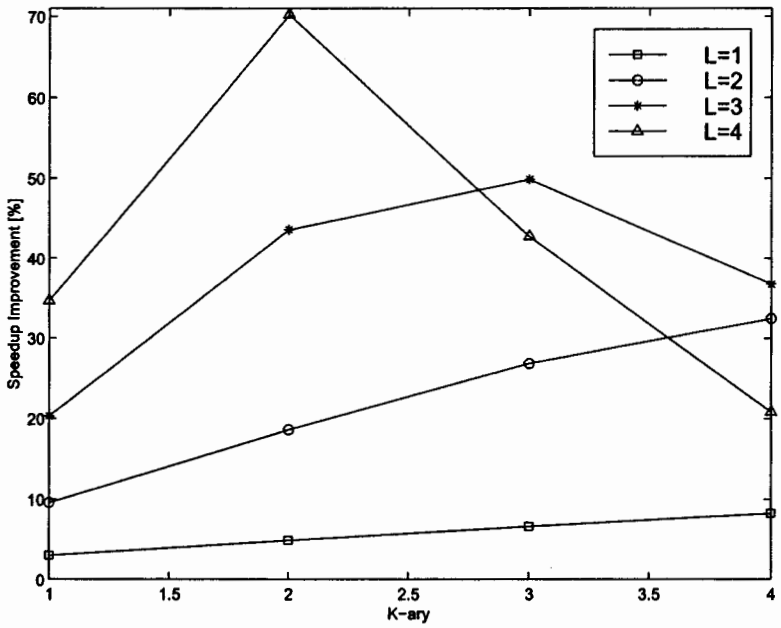

Figure 13: Speedup improvement in percentage vs. K and L; comparing equal division scheduling with multi-installment optimal scheduling; $w_i = 1$, $z_i = 0.05$, $T_{cp} = 1$, $T_{cm} = 1$, $T_{cm}^{sol} = 0.2$.