STONY BROOK UNIVERSITY

CEAS Technical Report 810

Scheduling Nonlinear Computational Loads

Jui Tsun Hung and Thomas G. Robertazzi

April 12, 2004

Scheduling Nonlinear Computational Loads

Jui Tsun Hung and Thomas G. Robertazzi

Abstract

It is demonstrated that supra-linear (greater than linear) speedup is possible in processing distributed divisible computational loads when computation time is a nonlinear function of load size. This result is radically different from the traditional distributed processing of divisible computational loads with linear processing complexity appearing in over 50 journal papers.

Index Terms

Scalability, Divisible load scheduling, Store and forward switching, Multilevel tree network, Nonlinear Computational Loads.

1 Introduction

Divisible loads are data parallel loads that are perfectly partitionable amongst links and processors. Such loads arise in the parallel and data intensive processing of massive amounts of data in grid computing, signal processing, image processing and experimental data processing. Since 1988 [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], work by a number of researchers has developed algebraic means of determining the optimal fractions of total load to assign to processors and

Thomas G. Robertazzi, a Senior Member, IEEE, Cosine Laboratory, Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. Phone (631) 632-8400, Fax (631) 632-8494, E-mail: tom@ece.sunysb.edu. The support of NSF grant CCR-99-12331 is acknowledged.

Jui Tsun Hung, Cosine Laboratory, Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. Phone (631) 632-8424, E-mail: trent@ece.sunysb.edu

links in a given interconnection topology under a given scheduling policy. Here optimality is defined in terms of speedup and solution time. The theory to date involves loads of linear computational complexity. That is, computation and communication time is proportional to the size of the load fraction assigned to a processor or link, respectively. With the right scheduling policy linear speedup in the number of processors can be achieved [4]. Here speedup is the ratio of solution time on one processor to solution time on N processors and is thus a measure of achievable parallel processing advantage.

The majority of the divisible load scheduling literature has appeared in computer engineering periodicals. Divisible load modeling should be of interest as it models, both computation and network communication in a completely integrated manner. Moreover, it is tractable with its linearity assumption. Optimal divisible load scheduling has been developed for various interconnection topologies [12], such as linear daisy chains [2], buses [6], trees [5], [13], hypercubes [7], and two and three dimensional meshes [14], [15]. A number of scheduling policies have been investigated including multi-installments [16] and multi-round scheduling [9], simultaneous distribution [4], [11], simultaneous start [10], detailed parameterizations, and solution time optimization [19], combinatorial schedule optimization [17]. Divisible loads may be divisible in fact or as an approximation as in the case of a large number of relatively small independent tasks [8]. Introductions to divisible load scheduling theory appear in [1], [3], [18].

In this paper we consider situations where the computational complexity of processing divisible load is a nonlinear function of the load size. It is shown, for tree networks, that if there is an (integer) χth power dependency of computation time at a node to the amount of load allocated to the node, one can solve for the optimal nodal load allocation by solving an χth order algebraic equation. For the special case of a single level tree (star) topology with certain scheduling policy assumptions, a closed form solution for an arbitrary integer power dependency can be found. Moreover, a recursive solution is possible for the case of a multilevel tree with power 2 (square law) dependency. Because of such nonlinear dependencies, supra-linear speedup is possible when load is distributed among multiple processors for concurrent processing.

We make three major assumptions in this paper. First, the computing and communication loads are

divisible (i.e. perfectly divisible with no precedence constraints [3]). Second, computation time is proportional to a nonlinear function of the size of the problem and transmission time is proportional to the size of the problem. Finally, each node transmits load concurrently (simultaneously) to its children.

A theorem is also introduced in this paper to categorize the nature of solutions when load has a nonlinear computational complexity. The theorem shows that in a practical problem to obtain an exact solution when processing nonlinear computational load in a distributed manner, post-processing to combine partial solutions is necessary. This categorization theorem is important for framing the context in which nonlinear divisible processing can be done. These novel results are radically different from the analysis of divisible loads with linear computational complexity appearing in over 50 journal papers [1].

This paper is organized as follows. The model description and notation appears in Section 2 of this paper. The classification of distribution loads is described in Section 3. In Section 4 and Section 5, the scheduling of heterogeneous single level tree and homogeneous multilevel tree using store and forward switching, simultaneous distribution, and staggered start protocols are first discussed (staggered start means that load can not be processed at a node until the node has completely received its quota of load). Here, without loss of generality, a power 2 relationship between the computing time and the amount of load is considered. The scheduling of trees using store and forward switching, simultaneous distribution, and simultaneous start (load can be processed as soon as a node has begun to receive its initial data load) and with a power χ (χ can be a real number) relationship for single level heterogeneous trees and a power 2 relationship for multilevel homogeneous trees is considered in Section 6 and Section 7. The conclusion is stated in Section 8.

2 Model, Notation and Theorem

2.1 Model and Notation for a Single Level Tree

In this paper two types of start models for each node in the multilevel tree network are presented. One is staggered start, the other is simultaneous (or concurrent) start. If a node begins to process its load after

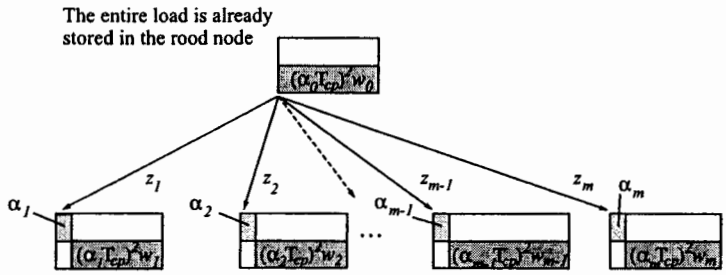

Fig. 1. A single level tree using staggered start with nonlinear load distribution.

the load is completely received, the protocol is called staggered start. If a node begins to process its load as soon as the load begins to be received, the protocol is called simultaneous start. Simultaneous start was proposed by Kim [10]. In both cases simultaneous distribution, where load is transmitted concurrently over multiple links, is used. Simultaneous distribution was first proposed by Piriyakumar and Murthy [11]. A single level subtree using staggered start is illustrated in Fig. 1 where each node contains a miniature timing diagram.

For a heterogeneous single level tree, which can be collapsed into an equivalent node, the notation is presented as follows:

α_0 : The load fraction assigned to the root processor.

α_i : The load fraction assigned to the ith link-processor pair.

w_i : The inverse computing speed of the ith processor.

w_{eq} : The inverse computing speed of an equivalent node representing a single level tree.

z_i : The inverse link speed of the ith link.

T_{cp} : Computing intensity constant. The entire load can be processed in $w_i T_{cp}^{\chi}$ seconds on the ith processor for a χth law dependency.

T_{cm} : Communication intensity constant. The entire load can be transmitted in $z_i T_{cm}$ seconds over the ith link.

$T_{f,m}$: The finish time of an equivalent node representing a single level tree composed of one root node

and m children nodes.

$T_{f,0}$: The finish time of a single processor only (i.e. a tree consisting of only a root node).

Definition 1: γ_{eq}, the ratio of the inverse computing speed on an equivalent node to that on the root node.

$$\gamma_{eq} = w_{eq}/w_0 \tag{1}$$

Definition 2: Speedup, the ratio of finish time on one processor (i.e. the root node) to that on an equivalent node representing a single level tree. This value is equal to the ratio of the inverse computing speed on the root node to that on an equivalent node, i.e. the inverse of γ_{eq}. Hence,

$$Speedup = T_{f,0}/T_{f,m} = w_0/w_{eq} = 1/\gamma_{eq} \tag{2}$$

Finally, $(\alpha_i T_{cp})^{\chi} w_i$ is the finish time to process the fraction α_i of the entire load on the ith processor.

2.2 Model and Notation for a Multilevel Tree

A heterogeneous multilevel tree network is too complicated to obtain a closed form solution of speedup. Therefore, a homogeneous multilevel tree network where root processors are equipped with a front-end processor for off-loading communications is evaluated. Root nodes, called intelligent roots, process a fraction of the load immediately while they start transmitting data to their children (see Fig. 2). After a sub-root receives all the assigned fraction of load for its descendants, it starts distributing these loads to its descendants immediately and concurrently. This strategy is called "store and forward switching with simultaneous distribution". Under store and forward switching load must be completely received by a node before being distributed to its descendants. The use of cut through switching is considered in another paper [20].

In the multilevel tree the bottom level is level 1 and the top level is level k. *The notation for a multilevel homogeneous fat tree* is denoted as follows.

$\alpha_{j,0}$: The load fraction assigned to a root processor of a jth level subtree.

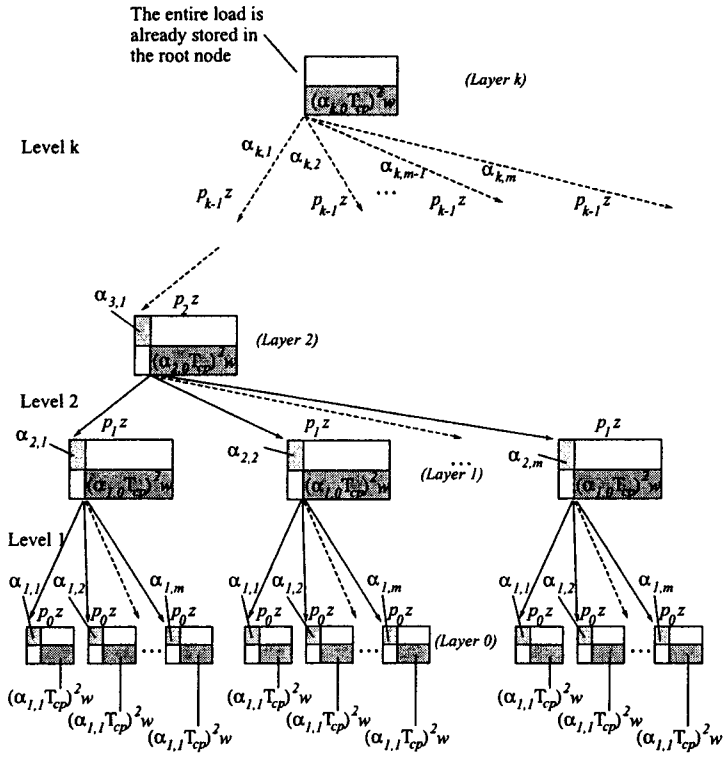

Fig. 2. Structure of multilevel homogeneous tree with store and forward switching, simultaneous distribution, staggered start.

$\alpha_{j,i}$: The load fraction assigned to the ith link-processor pair of a jth level subtree.

$w_{i_{eq_{j-1}}}$: The inverse computing speed of an equivalent ith node representing the $(j-1)$th level subtree, consisting of level $j-1$ descending to level 1. In a homogeneous multilevel tree, we assume that $w_{eq_{j-1}} = w_{i_{eq_{j-1}}}$ $(i = 1, 2, \ldots, m)$.

$T_{f,m}^{h,k}$: The finish time of a k level homogeneous tree with one root node and m equivalent children nodes.

Definition 3: $p_{j-1,i}$, the multiplier of the inverse capacity of the ith link at level j (see Fig. 2).

The value of the multiplier $p_{j-1,i}$ is defined here as the inverse of the total number of children processor descendants at and below level j for the ith subtree. The variable $p_{j-1,i}$ allows fat tree modeling. A fat tree allocates more capacity to nodes near the root to improve the transmission speed. In a homogeneous

multilevel fat tree, $p_{j-1} = p_{j-1,i}$ $(i = 1, 2, \ldots, m)$. We thus consider

$$p_{j-1} = \left(\sum_{l=0}^{j-1} m^l \right)^{-1} \qquad 0 < p_{j-1} \leq 1 \tag{3}$$

This choice of p_{j-1} allows an equivalent data rate of $1/z$ to each node in the tree from the root.

Definition 4: γ_j, the ratio of the inverse computing speed on an equivalent node at level j to that on the root node.

$$\gamma_j = w_{eq_j}/w \tag{4}$$

Definition 5: Speedup, the ratio of finish time on one processor (i.e. the root node) to that on an equivalent node representing a subtree from level k to level 1. This value is also equal to the ratio of the inverse computing speed on the root node to that on an equivalent node, i.e. the inverse of γ_k. Hence,

$$Speedup = T_{f,0}/T_{f,m}^{h,k} = w/w_{eq_k} = 1/\gamma_k \tag{5}$$

3 Classification of Distribution Loads

For parallel scheduling, the types of parallelism are function parallelism, and data parallelism. In this paper, the scheduling method is based on arbitrarily divisible data parallel loads. Loads can be classified as indivisible loads and divisible loads. Indivisible loads must be processed in a single processor for each load. They are data independent and can not be further partitioned into smaller sizes. Relevant scheduling methods can employ combinatorial and graph concepts [21], [22], [23].

Divisible loads can be identified as modularly divisible loads or arbitrarily divisible loads. Modularly divisible loads are partitioned into modules in advance and the associated data might be processed by different algorithms in different processors. However, arbitrarily divisible loads can be partitioned arbitrarily and the associated data should be processed by the same algorithm. Those fractions of partitioned data may or may not have the precedence relations.

3.1 Arbitrarily Divisible and Independent Loads

If an arbitrarily divisible load has data elements which are processed independently, the computing time is a linear function in terms of its load fraction size. Because of its tractable properties in linear analysis and calculation, divisible load scheduling theory has been developed and applied in many areas associated with massive amounts of data.

For example, in JPEG digital image processing, a block of 8×8 pixels subimage is considered as an independent element to be processed in a load. The entire load (one picture) is composed of many blocks. The computation and communication complexity function is a linear function of its load size (proportional to the number of elements).

Hence, for an arbitrarily divisible load the amount of computing is proportional to the number of elements in each fraction of load. The data in an element are processed independently. Each element has no processing relationship to other elements. This type of scheduling with a linear computation relationship has been developed extensively in the literature.

3.2 Arbitrarily Divisible and Dependent Loads

With nonlinear computational complexity there is some dependence of data elements on one another in terms of processing. Since the elements are dependent on one another, post-processing is needed (as in sorting algorithms, which are described later).

Some thought and knowledge of existing algorithms leads to the following theorem.

Theorem 1: For nonlinear divisible computational loads, (dependent loads), it is not possible to arbitrarily partition a load, do independent processing and both decrease solution time in a nonlinear manner and produce a solution *exactly* the same as that in a single processor. Either the solution is approximate, post-processing is necessary to combine partial solutions or both.

Proof: Let an entire load be a nonempty data set $\boldsymbol{S}$ consisting of n elements. A partition of a nonempty set $\boldsymbol{S}$ is a collection of nonempty subsets that are disjoint and whose union is $\boldsymbol{S}$ (see [24],

page 106). After being partitioned into $m+1$ subsets, $S_0, S_1, S_2, \ldots, S_m$, the entire load S is the union of the $m+1$ subsets. Provided that each processing step (instruction) takes the same time under the same computing capability for a specific algorithm, finish time will be proportional to the number of steps. Without loss of generality, let a load composed of n elements takes n^2 steps, then a function, $F(n) = n^2(steps)$, can be defined. Here, F is a virtual machine with n elements input and $n^2 steps$ output. For a fraction, α, of load of n elements, the number of processing steps, $F(\alpha n)$, is as following,

$$F(\alpha n) = (\alpha n)^2(steps) = \alpha^2 n^2 = \alpha^2 F(n) \tag{6}$$

where α is a fractional number.

Let a fraction of load, α, be partitioned into fractions, α_1 and α_2. That is,

$$F(\alpha n) = (\alpha n)^2 = \alpha^2 n^2 = \alpha^2 F(n) = (\alpha_1 + \alpha_2)^2 n^2 = (\alpha_1 + \alpha_2)^2 F(n) \tag{7}$$

$$= \alpha_1^2 F(n) + \alpha_2^2 F(n) + 2\alpha_1\alpha_2 F(n) = F(\alpha_1 n) + F(\alpha_2 n) + 2\alpha_1\alpha_2 F(n) \tag{8}$$

$$= F(\alpha_1 n) + F(\alpha_2 n) + \text{post-processing} \tag{9}$$

If the entire load is partitioned into $m+1$ subsets, and the fraction of each load is assigned as $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m$, the number of steps required is

$$F(n) = (n)^2 = [(\alpha_0 + \alpha_1 + \cdots + \alpha_m)n]^2 = (\alpha_0 + \alpha_1 + \cdots + \alpha_m)^2 \times n^2 \tag{10}$$

$$= (\alpha_0 + \alpha_1 + \cdots + \alpha_m)^2 \times F(n) = \left[\sum_{i=0}^{m} \alpha_i^2 + 2 \sum_{\substack{i=0, j=0 \\ i \neq j}}^{m} \alpha_i \cdot \alpha_j\right] \times F(n) \tag{11}$$

$$= \sum_{i=0}^{m} F(\alpha_i n) + 2 \sum_{\substack{i=0, j=0 \\ i \neq j}}^{m} \alpha_i \cdot \alpha_j \times F(n) \tag{12}$$

Observing the above equation, the first term is a summation of the amount of processing steps for each fraction processed by a child node, and the second term is the amount of steps for post-processing. Therefore, post-processing is necessary for an exact solution if the processing steps is a nonlinear function of the number of elements in a load using a specific algorithm. □

Corollary 1: Nonlinear speedup improvement is possible, but solution time for divisible processing must include partitioned load solution time and post-processing time.

Proof: The following application demonstrates this. □

3.3 An Application

An example application where the computation time to process divisible load is nonlinear and a nonlinear processing time advantage can be realized is now presented.

Sorting lists of discrete elements is a classic combinatorial problem and is widely performed. It is well known that sorting a list of N elements (say alphabetically) has a minimal computational time complexity of $O(NlogN)$ (versus $O(N^2)$ for naive sorting). As an example, consider a list of N items as Fig. 3, which are split into three sublists, each of which is placed on one of three discrete processors. Each

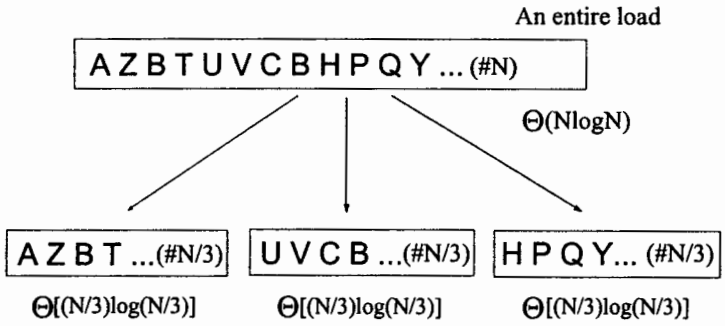

Fig. 3. A load with N elements is partitioned into three equal fractions. All are processed by the same sorting algorithm.

sublist on a processor is sorted. Since the three processors work concurrently and since each list has $N/3$ elements, the computational complexity of the overall sorting is $O((N/3)log(N/3))$.

Suppose now that we merge the first two sublists (Fig. 4). This requires $N/3$ comparison operations. To merge the newly created sublist with the third original sublist (see Fig. 5) requires $(2/3)N$ comparison operations. The overall computational complexity for sorting on three processors is then

$$\frac{N}{3}log\frac{N}{3} + N \tag{13}$$

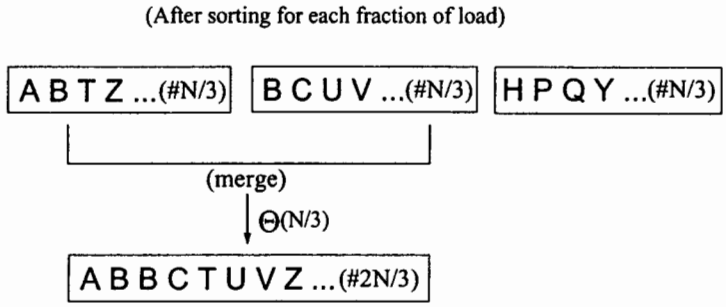

Fig. 4. Merging two fractional results, (or two subset results), to obtain a partial sorting solution.

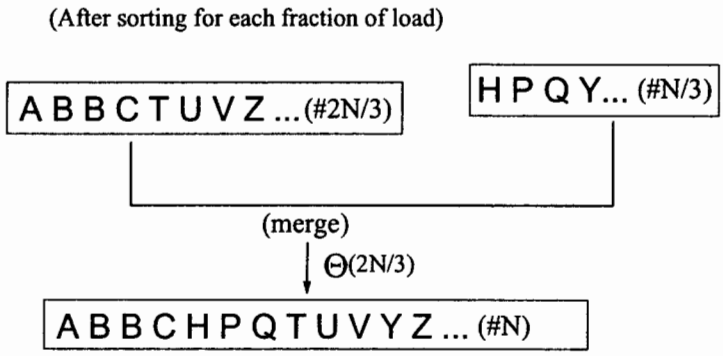

Fig. 5. Merging two fractional results, (or two subset results), to obtain the final sorting solution.

The first term here is the cost of distributed processing and the second term is the cost of post processing. This approach is superior to the computational cost of sorting the original list as $N \rightarrow \infty$. Note that whether divisible load processing results in a computational saving taking post processing into accounts is problem dependent.

Naturally there are communication costs in the above scenario. However the same decomposed computation on a single sequential processor would also result in a computational time savings, though with somewhat different constants. Note that if a list was broken into M, ($M > 3$), sublists, concurrent merging would lower the value of the computational complexity.

4 Processors Using Staggered Start in a Single Level Tree

In this section we consider a heterogeneous single level tree using store and forward switching and simultaneous distribution in a single level tree. Processors use the staggered start protocol (in contrast to simultaneous start) to process their load. In the staggered start protocol a processor must receive its load completely before it begins to process the load. The root node is assumed to be an "intelligent" node, so it can distribute load to its children while processing some fraction of the load. In this sense the root may be considered to have a front end sub-processor for communications off-loading, while the children do not [2], [3]. We will discuss a homogeneous multilevel tree in the next section.

4.1 Single Level Tree: Root Node with Data Storage, Power 2

The structure of a single level tree network with intelligent root, $m+1$ processors and m links is illustrated in Fig. 1.

All children processors are connected to the root processor via direct communication links. The intelligent root processor, assumed to be the only processor at which the divisible load arrives, partitions a total processing load optimally into $m+1$ fractions, keeps its own fraction, α_0, and distributes the other fractions, $\alpha_1, \alpha_2, \ldots, \alpha_m$, to the children processors respectively and concurrently. Without loss of generality, it is assumed that computation time is quadratic in load size and communication time is linear in load size.

After completely receiving all of its assigned fraction of load, each processor begins computing immediately and continues without any interruption until all of its assigned load fraction has been processed. Staggered start models a computer which must receive all incoming load before processing its fraction. In order to minimize the processing finish time, all of the utilized processors in the network must finish computing at the same time [3]. The process of load distribution can be represented by Gantt chart-like timing diagrams, as illustrated in Fig. 6. Note that this is a completely deterministic model.

It is assumed that all load is available for distribution from the root node at time $t = 0$. This root

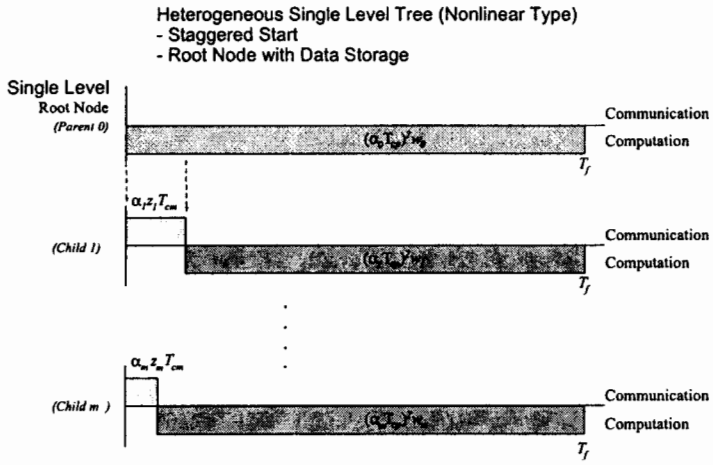

Fig. 6. Timing diagram of single level tree with simultaneous distribution, staggered start, and root node with data storage.

node with data storage model will be used for the level k subtree later when examining multilevel trees. According to the timing diagram Fig. 6, the fundamental recursive equations of the system can be formulated as follows:

$$(\alpha_0 T_{cp})^2 w_0 = \alpha_i z_i T_{cm} + (\alpha_i T_{cp})^2 w_i \qquad i = 1, 2, \ldots, m \tag{14}$$

The normalization equation for the single level tree with intelligent root is

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{15}$$

This yields $m + 1$ linear equations with $m + 1$ unknowns. Note that this is a completely deterministic model. Manipulating the recursive equations and normalization equation can yield the solution for the fractions of load distribution. From (14),

$$(\alpha_i T_{cp})^2 w_i + \alpha_i z_i T_{cm} = (\alpha_0 T_{cp})^2 w_0 \tag{16}$$

$$(\alpha_i)^2 + \frac{z_i T_{cm}}{w_i T_{cp}^2} \alpha_i - \frac{w_0 T_{cp}^2}{w_i T_{cp}^2} \alpha_0^2 = 0 \tag{17}$$

Let

$$\varsigma_i = \frac{z_i T_{cm}}{w_i T_{cp}^2} = \sigma_i \cdot \frac{1}{T_{cp}} \qquad i = 1, 2, \ldots, m \tag{18}$$

where

$$\sigma_i = \frac{z_i T_{cm}}{w_i T_{cp}} \qquad i = 1, 2, \ldots, m \tag{19}$$

and

$$\xi_i = \frac{w_0 T_{cp}^2}{w_i T_{cp}^2} = \frac{w_0}{w_i} \qquad i = 1, 2, \ldots, m \tag{20}$$

Hence, (17) becomes

$$(\alpha_i)^2 + \varsigma_i \alpha_i - \xi_i \alpha_0^2 = 0 \tag{21}$$

Applying the quadratic equation to (21), one obtains

$$\alpha_i = \frac{-\varsigma_i \pm \sqrt{\varsigma_i^2 + 4\xi_i \alpha_0^2}}{2 \times 1} \tag{22}$$

Since the value of α_i is the fraction of load, it does not make any physical sense if $\alpha_i < 0$. Hence, $\alpha_i \geq 0$. Consequently,

$$\alpha_i = \frac{-\varsigma_i + \sqrt{\varsigma_i^2 + 4\xi_i \alpha_0^2}}{2} \qquad i = 1, 2, \ldots, m \tag{23}$$

According to (23), the normalization equation (15) becomes

$$\alpha_0 + \sum_{i=1}^{m} \frac{-\varsigma_i + \sqrt{\varsigma_i^2 + 4\xi_i \alpha_0^2}}{2} = 1 \tag{24}$$

The value of α_0 can be obtained by using the quadratic equation to solve (24). We assume the solution of α_0 is C_0, a specific value. Then we can apply this value to (23) and obtain the values of distribution fractions of load α_i as follows:

$$\alpha_i = \frac{-\varsigma_i + \sqrt{\varsigma_i^2 + 4\xi_i C_0^2}}{2} \tag{25}$$

According to Fig. 6, the finish time is achieved as

$$T_{f,m} = (\alpha_0 T_{cp})^2 w_0 = (C_0 T_{cp})^2 w_0 \tag{26}$$

The term, $T_{f,m}$, indicates the finish time for each single divisible load completed in a single level tree consisting of one root node as well as m children nodes. Another term, $T_{f,0}$, is the finish time for the

entire divisible load processed only on the root processor; that is, $T_{f,0}$ is the finish time for only one root node tree without any children nodes. This leads to

$$T_{f,0} = (\alpha_0 T_{cp})^2 w_0 = (1 \times T_{cp})^2 w_0 = T_{cp}^2 w_0 \tag{27}$$

Now, if we collapse the single level tree into an equivalent node, we can obtain the finish time, $T_{f,eq}$, in terms of the inverse of equivalent computing speed, w_{eq}, for the equivalent node as follows:

$$T_{f,eq} = (1 \times T_{cp})^2 w_{eq} = T_{cp}^2 w_{eq} \tag{28}$$

The finish time, $T_{f,eq}$, is the same as that in (26). Therefore, according to (26) and (28), we obtain

$$T_{cp}^2 w_{eq} = (C_0 T_{cp})^2 w_0 \tag{29}$$

According to Definition 1 in Section 2 (i.e. $\gamma_{eq} = w_{eq}/w_0 = T_{f,m}/T_{f,0}$), the value of γ_{eq} can be obtained from (29) as

$$\gamma_{eq} = C_0^2 = \alpha_0^2 \tag{30}$$

Speedup is the ratio of job solution time on one processor to job solution time on the $m+1$ processors (see Definition 2 in Section 2.) Thus,

$$Speedup = \frac{1}{\gamma_{eq}} = \frac{1}{C_0^2} = \left(\frac{1}{\alpha_0}\right)^2 \tag{31}$$

Speedup is a measure of the achievable parallel processing advantage.

1) *Homogeneous Case:* ($w_i = w$ and $z_i = z$ for $i = 1, 2, \ldots, m$.)

As a special case, consider the situation of a homogeneous network where all children processors have the same inverse computing speed and all links have the same inverse transmission speed. In other words, $w_i = w$ and $z_i = z$ for $i = 1, 2, \ldots, m$. Note that the root inverse computing speed, w_0, can be different from w_i, where $i = 1, 2, \ldots, m$.

From (18) and (20),

$$\varsigma_i = \frac{zT_{cm}}{wT_{cp}^2} = \varsigma = \sigma \cdot \frac{1}{T_{cp}} \qquad i = 1, 2, \ldots, m \tag{32}$$

where

$$\sigma = zT_{cm}/wT_{cp} \tag{33}$$

and

$$\xi_i = \frac{w_0 T_{cp}^2}{w T_{cp}^2} = \frac{w_0}{w} = \xi \qquad i = 1, 2, \ldots, m \tag{34}$$

Hence, (23) becomes

$$\alpha_i = \frac{-\varsigma + \sqrt{\varsigma^2 + 4\xi\alpha_0^2}}{2} \qquad i = 1, 2, \ldots, m \tag{35}$$

According to (35), the normalization equation (15) becomes

$$\alpha_0 + \sum_{i=1}^{m} \frac{-\varsigma + \sqrt{\varsigma^2 + 4\xi\alpha_0^2}}{2} = 1 \tag{36}$$

$$\alpha_0 + m \cdot \frac{-\varsigma + \sqrt{\varsigma^2 + 4\xi\alpha_0^2}}{2} = 1 \tag{37}$$

Manipulate (37) as follows:

$$2\alpha_0 - m\varsigma + m\sqrt{\varsigma^2 + 4\xi\alpha_0^2} = 2 \tag{38}$$

$$\left(m\sqrt{\varsigma^2 + 4\xi\alpha_0^2}\right)^2 = [(2 + m\varsigma) - 2\alpha_0]^2 \tag{39}$$

$$(m^2\xi - 1)\alpha_0^2 + (2 + m\varsigma)\alpha_0 - (m\varsigma + 1) = 0 \tag{40}$$

Applying the quadratic equation to (40), one obtains

$$\begin{aligned} \alpha_0 &= \frac{-(2 + m\varsigma) \pm \sqrt{(2 + m\varsigma)^2 + 4(m^2\xi - 1)(m\varsigma + 1)}}{2 \cdot (m^2\xi - 1)} \\ &= \frac{-(2 + m\varsigma) \pm \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}}{2(m^2\xi - 1)} \end{aligned} \tag{41}$$

Since α_0 is the fraction of load for computation at the root node, it does not make any physical sense if the value of α_0 is less than zero. Thus, in (41) sign $+$ is taken rather than $\pm$ so that the value of α_0 is greater than zero. Therefore the solution of α_0 is

$$\alpha_0 = \frac{-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}}{2(m^2\xi - 1)} \tag{42}$$

Substituting the value of α_0 in (35), we obtain the value of unknown fraction, α_i, where $i = 1, 2, \ldots, m$.

According to Fig. 6, the finish time in the homogeneous single level tree is

$$T^h_{f,m} = (\alpha_0 T_{cp})^2 w_0 \tag{43}$$

The value, $T^h_{f,m}$, indicates the finish time for a homogeneous single level tree, consisting of one root node as well as m children nodes.

Applying a similar derivation to that in heterogeneous single level tree, we obtain

$$\gamma_{eq} = \alpha_0^2 = \left(\frac{-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}}{2(m^2\xi - 1)} \right)^2 \tag{44}$$

and

$$Speedup = \frac{1}{\gamma_{eq}} = \left(\frac{2(m^2\xi - 1)}{-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}} \right)^2 \tag{45}$$

Speedup is a measure of the achievable parallel processing advantage.

2) *Special Case:* $\sigma \ll 1$ and $\lambda \gg 1$, i.e. $\varsigma \ll 1$ (fast communication).

Repeat (32) here as follows:

$$\varsigma = \frac{zT_{cm}}{wT^2_{cp}} = \sigma \cdot \frac{1}{T_{cp}} = \frac{\sigma}{\lambda} \qquad \text{assume} \quad \lambda = T_{cp} \tag{46}$$

Provided that communication is faster than processing, then $\sigma \ll 1$. If we also assume $\lambda \gg 1$, we can conclude that $\varsigma \ll 1$. Since $m\varsigma = m\sigma/\lambda$, if one assume $m \ll \lambda$, then $m\varsigma \ll 1$. Therefore, $m\varsigma + 1 \to 1$ and $2 + m\varsigma \to 2$. The speedup formula (45) can be approximated as

$$Speedup = \left(\frac{2(m^2\xi - 1)}{-2 + \sqrt{m^2\varsigma^2 + 4m^2\xi}} \right)^2 \tag{47}$$

Now, we compare $4m^2\xi$ with $m^2\varsigma^2$ by the following equation (48).

$$4m^2\xi - m^2\varsigma^2 = 4m^2\xi - m^2\frac{\sigma^2}{\lambda^2} = 4m^2 \cdot \frac{w_0}{w} - m^2\sigma^2 \cdot \frac{1}{\lambda^2} \tag{48}$$

and assume $\xi = w_0/w$ (see (34)) is about one (all processors have about the same computation speed) or is greater than one, then $4m^2\xi \gg 4$. Since λ is a large number and $\sigma \ll 1$, then $\sigma^2/\lambda^2 \ll 1$.

In other words, $\varsigma^2 \ll 1$. This leads to

$$4m^2\xi \gg m^2\varsigma^2 \tag{49}$$

Therefore, $4m^2\xi + m^2\varsigma^2 \rightarrow 4m^2\xi$. The speedup (47) becomes

$$Speedup = \left(\frac{2(m^2\xi - 1)}{-2 + \sqrt{4m^2\xi}}\right)^2 = \left(\frac{2(m^2\xi - 1)}{2(m\sqrt{\xi} - 1)}\right)^2 = \left(m\sqrt{\xi} + 1)\right)^2 \tag{50}$$

3) *Specific Case:* $\xi = 1$.

If the computing capability of the root node is the same as that of the children nodes for a homogeneous single level tree, i.e. $w_0 = w$, then $\xi = 1$. Under such condition, the speedup formula becomes

$$Speedup = (m + 1)^2 \tag{51}$$

This makes intuitive sense if communication is much faster than computation.

5 Processors Using Staggered Start Protocol in a Homogeneous Multilevel Fat Tree Analysis

A fat tree architecture is now considered where upper links have more capacity than lower links in such a way that each node has equivalent bandwidth $1/z$ to the root. Properly designed fat trees preclude any tree level from becoming a capacity bottleneck. Such an architecture will allow a maximization of performance. Consider a homogeneous multilevel fat tree network where all processors have the same inverse computing speed, w, and links of level j have the transmission speed, $p_{j-1}z$, (see Fig. 2). The value of p_{j-1} is defined on Definition 3 in Section 2.

In this work store and forward switching (in contrast to cut through switching) from level to level is studied. According to store and forward switching, load must be completely received by a nodes before being distributed to its descendants. The process of load distribution for a multilevel fat tree network using store and forward switching from upper level to lower level can be represented by a Gantt chart-like timing diagram, (see Fig. 7). We will derive the speedup of the whole multilevel tree by moving

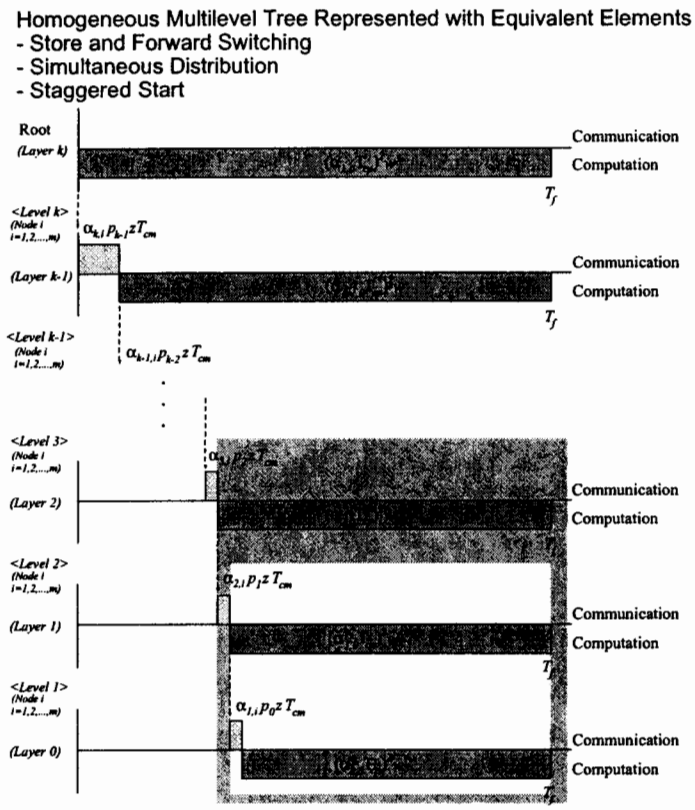

Fig. 7. Timing diagram of a homogeneous multilevel tree with store and forward switching, simultaneous distribution, staggered start, root node with data storage. Subtrees are collapsed into equivalent nodes from bottom most level to topmost level.

upwards through the tree, collapsing successive subtrees into equivalent processors until the entire single level tree is collapsed into an equivalent node. We find that each "box" (level) in Fig. 7 illustrates the scheduling levels of a multilevel tree where the root node has data storage (all load is available at the single level tree root at time $t = 0$). The nested, shaded boxes indicate single level trees which are collapsed into equivalent nodes.

5.1 Level j Subtree: Root Node with Data Storage

As in Fig. 2, let level k be the top root single level subtree. Here level "j" is used to represent any single level subtree at any arbitrary level j. Let $\alpha_{j,i}$ be the load fraction for the ith children collapsed (or

equivalent) node of the jth level subtree. It is also assumed that

$$(\alpha_{j,i}T_{cp})^2 w_{eq_{j-1}} > \alpha_{j,i} p_{j-1} z T_{cm} \tag{52}$$

in this subtree (see Fig. 8). That is, communication time is faster than computation time. According to Fig. 8, the fundamental recursive equations of the jth level subtree network are

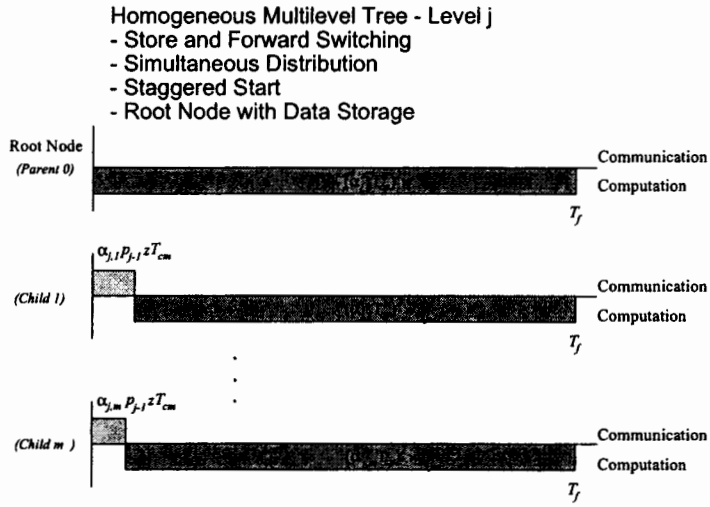

Fig. 8. Timing diagram of jth level subtree with simultaneous distribution, staggered start, and root node with data storage.

$$(\alpha_{j,0}T_{cp})^2 w = (\alpha_{j,i}T_{cp})^2 w_{eq_{j-1}} + \alpha_{j,i} p_{j-1} z T_{cm} \qquad j = 1, 2, \ldots, k \quad \text{and} \quad i = 1, 2, \ldots, m \tag{53}$$

The normalization equation for the jth single level subtree with intelligent root is

$$\alpha_{j,0} + \alpha_{j,1} + \alpha_{j,2} + \cdots + \alpha_{j,m} = 1 \tag{54}$$

This yields $m+1$ linear equations with $m+1$ unknowns. From (53), one has:

$$\alpha_{j,i}^2 + \frac{p_{j-1} z T_{cm}}{w_{eq_{j-1}} T_{cp}^2} \alpha_{j,i} - \frac{w T_{cp}^2}{w_{eq_{j-1}} T_{cp}^2} \alpha_{j,0}^2 = 0 \tag{55}$$

$$\alpha_{j,i}^2 + p_{j-1} \frac{w}{w_{eq_{j-1}}} \times \frac{z T_{cm}}{w T_{cp}^2} \alpha_{j,i} - \frac{w}{w_{eq_{j-1}}} \alpha_{j,0}^2 = 0 \tag{56}$$

Referring to Definition 4 in Section 2, (i.e. $\gamma_{j-1} = w_{eq_{j-1}}/w$), (33) (i.e. $\sigma = zT_{cm}/wT_{cp}$), and (46) (i.e. $\varsigma = \sigma \cdot 1/T_{cp}$), equation (56) becomes

$$\alpha_{j,i}^2 + \frac{p_{j-1}}{\gamma_{j-1}} \varsigma \cdot \alpha_{j,i} - \frac{1}{\gamma_{j-1}} \alpha_{j,0}^2 = 0 \tag{57}$$

Applying the quadratic equation to (57), one obtains the solution of $\alpha_{j,i}$ as follows:

$$\alpha_{j,i} = \frac{-\frac{p_{j-1}}{\gamma_{j-1}}\varsigma \pm \sqrt{\left(\frac{p_{j-1}}{\gamma_{j-1}}\varsigma\right)^2 + \frac{4}{\gamma_{j-1}}\alpha_{j,0}^2}}{2} \tag{58}$$

Since $\alpha_{j,i} > 0$ (the same reason as before), we obtain the final solution of $\alpha_{j,i}$ as

$$\alpha_{j,i} = \frac{-\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + \sqrt{\left(\frac{p_{j-1}}{\gamma_{j-1}}\varsigma\right)^2 + \frac{4}{\gamma_{j-1}}\alpha_{j,0}^2}}{2} \qquad i = 1, 2, \ldots, m \tag{59}$$

The fraction of distribution load, $\alpha_{j,1}$, can be solved by employing the normalization equation (54). According to (59), (54) becomes

$$\alpha_{j,0} + \sum_{i=1}^{m} \alpha_{j,i} = 1 \tag{60}$$

$$\alpha_{j,0} + \sum_{i=1}^{m} \frac{-\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + \sqrt{\left(\frac{p_{j-1}}{\gamma_{j-1}}\varsigma\right)^2 + \frac{4}{\gamma_{j-1}}\alpha_{j,0}^2}}{2} = 1 \tag{61}$$

$$\left(\frac{m^2}{\gamma_{j-1}} - 1\right)\alpha_{j,0}^2 + \left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 2\right)\alpha_{j,0} - \left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 1\right) = 0$$

Therefore, using the quadratic equation, one obtains

$$\alpha_{j,0} = \frac{1}{2\left(\frac{m^2}{\gamma_{j-1}} - 1\right)} \times \left\{ -\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 2\right) \pm \sqrt{\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma\right)^2 + 4\frac{m^2}{\gamma_{j-1}}\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 1\right)} \right\} \tag{62}$$

Since $\alpha_{j,0} > 0$ (the same reason as before), one obtains

$$\alpha_{j,0} = \frac{1}{2\left(\frac{m^2}{\gamma_{j-1}} - 1\right)} \times \left\{ -\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 2\right) + \sqrt{\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma\right)^2 + 4\frac{m^2}{\gamma_{j-1}}\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 1\right)} \right\} \tag{63}$$

The equivalent finish time, $T_{f,m}^{h,j}$, for a homogeneous jth level subtree with m children nodes is

$$T_{f,m}^{h,j} = (\alpha_{j,0} T_{cp})^2 w \tag{64}$$

Also,

$$T_{f,m}^{h,j} = (1 \cdot T_{cp})^2 w_{eq_j} = (\alpha_{j,0} T_{cp})^2 w \tag{65}$$

Consequently, we obtain

$$\gamma_j = \frac{w_{eq_j}}{w} = (\alpha_{j,0})^2 = \frac{1}{\left[2\left(\frac{m^2}{\gamma_{j-1}} - 1\right)\right]^2} \times \left\{ \left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 2 \right)^2 + \left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma \right)^2 + 4 \times \frac{m^2}{\gamma_{j-1}} \left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 1 \right) - 2\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 2 \right) \times \sqrt{\left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma \right)^2 + 4 \times \frac{m^2}{\gamma_{j-1}} \left(m\frac{p_{j-1}}{\gamma_{j-1}}\varsigma + 1 \right)} \right\} \tag{66}$$

and

$$Speedup = \frac{1}{\gamma_j} \qquad j = 1, 2, \ldots, k \tag{67}$$

For a *homogeneous multilevel fat tree*, if the computation capability of each node is w, the value of w_{eq_0} is equal to w, and then the initial value of γ_0 can be derived as follows:

$$\gamma_0 = \frac{w_{eq_0}}{w} = \frac{w}{w} = 1 \tag{68}$$

If $p_{j-1}\varsigma$ approaches zero, (large tree where communication is much faster than computation), the model approaches an ideal case. Each node can receive the load instantly and compute the data immediately. Under such assumption, the recursive function (66) can be approximated as

$$\gamma_j = \frac{(2)^2 + 4\frac{m^2}{\gamma_{j-1}} - 2(2)\sqrt{4\frac{m^2}{\gamma_{j-1}}}}{\left[2\left(\frac{m^2}{\gamma_{j-1}} - 1\right)\right]^2} = \frac{1}{\left(\frac{m}{\sqrt{\gamma_{j-1}}} + 1\right)^2} \qquad j = 1, 2, \ldots, k \tag{69}$$

According to $\gamma_0 = 1$ and (69), the recursive formulae are obtained as follows.

$$\gamma_1 = \frac{1}{(m+1)^2} \tag{70}$$

$$\gamma_2 = \frac{1}{\left(\frac{m}{\sqrt{\frac{1}{(m+1)^2}}} + 1\right)^2} = \frac{1}{(m^2 + m + 1)^2} \tag{71}$$

$$\vdots$$

$$\gamma_j = \frac{1}{(m^j + m^{j-1} + \cdots + m + 1)^2} = \frac{1}{(\sum_{l=0}^{j} m^l)^2} \qquad j = 1, 2, \ldots, k \tag{72}$$

Consequently, speedup becomes

$$Speedup = \frac{1}{\gamma_k} = (\sum_{l=0}^{k} m^l)^2 \tag{73}$$

We conclude that speedup is square to the total number of nodes, which is $m^0 + m^1 + m^2 + \cdots + m^k$, which makes intuitive sense.

6 Processors Using Simultaneous Start Protocol in a Heterogeneous Single Level Tree

In this section we discuss only the root node with data storage case and power χ model for a heterogeneous single level tree.

The process of load distribution can be represented by Gantt chart-like timing diagrams, as illustrated in Fig. 9. According to the timing diagram of Fig. 9, the fundamental recursive equations of the system

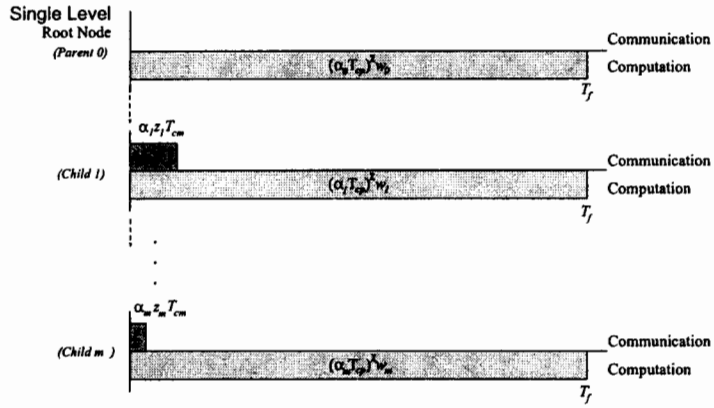

Fig. 9. Timing diagram of a single level tree with simultaneous distribution, simultaneous start, and root node with data storage.

can be formulated as follows:

$$(\alpha_0 T_{cp})^{\chi} w_0 = (\alpha_1 T_{cp})^{\chi} w_1 \tag{74}$$

$$(\alpha_{i-1} T_{cp})^{\chi} w_{i-1} = (\alpha_i T_{cp})^{\chi} w_i \qquad i = 2, \ldots, m \tag{75}$$

The normalization equation for the single level tree with intelligent root is

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{76}$$

This yields $m+1$ linear equations with $m+1$ unknowns. According the similar derivation to the Section 4, one obtains the value of speedup from $T_{f,0}/T_{f,m}$, which is equal to $1/\gamma_{eq}$. Thus,

$$Speedup = \frac{1}{\gamma_{eq}} = \left(\frac{1}{\alpha_0}\right)^{\chi} = w_0 \left(\sum_{l=0}^{m} \sqrt[\chi]{\frac{1}{w_l}}\right)^{\chi} \tag{77}$$

Speedup is a measure of the achievable parallel processing advantage.

7 Processors Using Simultaneous Start Protocol in a Homogeneous Multilevel Fat Tree

In this section a homogeneous multilevel tree using store and forward switching, simultaneous distribution, and simultaneous start is discussed. In this structure the topmost single level subtree is called level k and levels below it are generally level j (j goes from level 1 at the bottom most level up to level $k-1$). The speedup of the whole multilevel tree is obtained by successively collapsing single level subtrees into equivalent nodes until the entire tree is collapsed into an equivalent node (see Fig 10). A root without

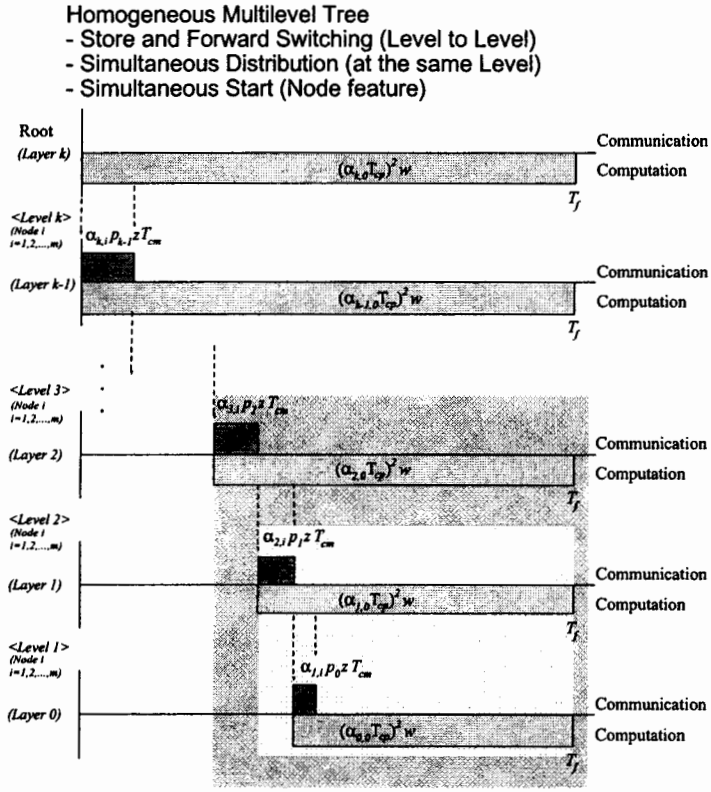

Fig. 10. Timing diagram of a homogeneous multilevel tree with store and forward switching, simultaneous distribution, simultaneous start. Subtrees are collapsed into equivalent nodes from bottommost level to topmost level.

data storage model is first applied to level j (i.e. $j = 1, 2, \ldots, k-1$), and then the root with data storage

model is used for the topmost level (level k). The latter choice reflects the need for nodes (except for the root) to receive load completely before they can forward it to their children.

7.1 Level j Subtree: Root Node without Data Storage, Power 2

According to Fig. 11, the fundamental recursive equations of the jth-level tree network are

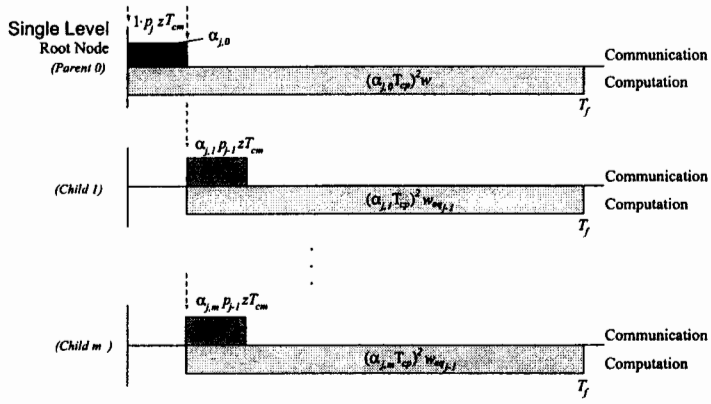

Fig. 11. Timing diagram of level j subtree using store and forward switching, simultaneous distribution, simultaneous start, and root node without data storage

$$(\alpha_{j,0}T_{cp})^2 w = (\alpha_{j,1}T_{cp})^2 w_{eq_{j-1}} + 1 \cdot p_j z T_{cm} \tag{78}$$

$$(\alpha_{j,i-1}T_{cp})^2 w_{eq_{j-1}} = (\alpha_{j,i}T_{cp})^2 w_{eq_{j-1}} \qquad i = 2, 3, \ldots, m \tag{79}$$

The normalization equation for the jth single level tree with intelligent root is

$$\alpha_{j,0} + \alpha_{j,1} + \alpha_{j,2} + \cdots + \alpha_{j,m} = 1 \tag{80}$$

This yields $m + 1$ linear equations with $m + 1$ unknowns.

Equation (78) becomes

$$\alpha_{j,1}^2 = \frac{wT_{cp}^2}{w_{eq_{j-1}}T_{cp}^2}\alpha_{j,0}^2 - \frac{p_j z T_{cm}}{w_{eq_{j-1}}T_{cp}^2} = \frac{w}{w_{eq_{j-1}}}\alpha_{j,0}^2 - \frac{p_j w}{w_{eq_{j-1}}} \times \frac{zT_{cm}}{wT_{cp}^2} \tag{81}$$

According to (32), $\varsigma = zT_{cm}/wT_{cp}^2$, and the derivative of Definition 4 in Section 2, $\gamma_{j-1} = w_{eq_{j-1}}/w$, (81) becomes

$$\alpha_{j,1}^2 = \frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - \frac{1}{\gamma_{j-1}}p_j\varsigma \tag{82}$$

Solving for $\alpha_{j,0}$ using the normalization equation as was done for the staggered start policy, one obtains

$$\alpha_{j,0} = \frac{-1+\sqrt{1+\left(1+m^2\frac{p_j\varsigma}{\gamma_{j-1}}\right)\left(\frac{m^2}{\gamma_{j-1}}-1\right)}}{\frac{m^2}{\gamma_{j-1}}-1} = \frac{-1+\sqrt{\frac{m^4}{\gamma_{j-1}^2}p_j\varsigma-\frac{m^2}{\gamma_{j-1}}p_j\varsigma+\frac{m^2}{\gamma_{j-1}}}}{\frac{m^2}{\gamma_{j-1}}-1} \tag{83}$$

Since $T_{f,m}^{h,j}$ is the finish time for a equivalent homogeneous jth-level subtree, one can obtain

$$T_{f,m}^{h,j} = (1\cdot T_{cp})^2 w_{eq_j} = (\alpha_{j,0}T_{cp})^2 w \qquad j=1,2,\ldots,k-1 \tag{84}$$

According to Definition 4 in Section 2,

$$\begin{aligned}\gamma_j =& \frac{w_{eq_j}}{w} = \alpha_{j,0}^2 \\ =& \frac{1}{\left(\frac{m^2}{\gamma_{j-1}}-1\right)^2}\times\left\{1+\frac{m^4}{\gamma_{j-1}^2}p_j\varsigma-\frac{m^2}{\gamma_{j-1}}p_j\varsigma+\frac{m^2}{\gamma_{j-1}}-2\sqrt{\frac{m^4}{\gamma_{j-1}^2}p_j\varsigma-\frac{m^2}{\gamma_{j-1}}p_j\varsigma+\frac{m^2}{\gamma_{j-1}}}\right\}\end{aligned} \tag{85}$$

where $j=1,2,\ldots,k-1$.

7.2 The Topmost kth-level Subtree: Root Node with Data Storage, Power 2

In this subsection the topmost level (level k) subtree is discussed. The timing diagram of the topmost equivalent subtree, level k, is Fig. 12. Accordingly, the fundamental recursive equations of the kth-level

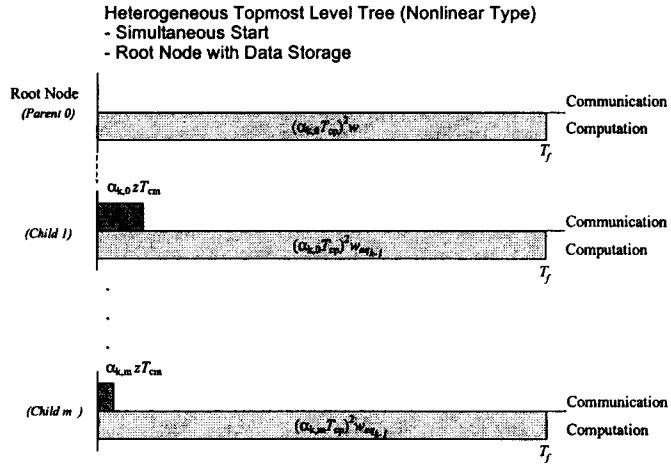

Fig. 12. Timing diagram of topmost level k subtree using store and forward switching, simultaneous distribution, simultaneous start, and root node with data storage

subtree can be derived as follows.

$$(\alpha_{k,0}T_{cp})^2 w = (\alpha_{k,i}T_{cp})^2 w_{eq_{k-1}} \quad i = 1, 2, 3, \ldots, m \tag{86}$$

The normalization equation for the kth single level tree with intelligent root is

$$\alpha_{k,0} + \alpha_{k,1} + \alpha_{k,2} + \cdots + \alpha_{k,m} = 1 \tag{87}$$

This gives $m + 1$ linear equations with $m + 1$ unknowns. According to (86),

$$\alpha_{k,i}^2 = \frac{wT_{cp}^2}{w_{eq_{k-1}}T_{cp}^2}\alpha_{k,0}^2 = \frac{1}{\gamma_{k-1}}\alpha_{k,0}^2 \tag{88}$$

$$\alpha_{k,i} = \pm\sqrt{\frac{1}{\gamma_{k-1}}}\alpha_{k,0} \tag{89}$$

Since $\alpha_{k,i} > 0$ (the same reason as before),

$$\alpha_{k,i} = \frac{1}{\sqrt{\gamma_{k-1}}}\alpha_{k,0} \qquad i = 1, 2, \ldots, m \tag{90}$$

Applying (90) to the normalization equation (87), we obtain the value of $\alpha_{k,0}$ as follows:

$$\alpha_{k,0} = \frac{1}{\frac{m}{\sqrt{\gamma_{k-1}}} + 1} \tag{91}$$

Therefore, the equivalent finish time, $T_{f,m}^{h,k}$, for a homogeneous kth level tree with m children nodes can be obtained. That is,

$$T_{f,m}^{h,k} = (1 \cdot T_{cp})^2 w_{eq_k} = (\alpha_{k,0}T_{cp})^2 w$$

From Definition 4 in Section 2,

$$\gamma_k = \frac{w_{eq_k}}{w} = \alpha_{k,0}^2 = \frac{1}{\left(\frac{m}{\sqrt{\gamma_{k-1}}} + 1\right)^2} \tag{92}$$

Finally, we obtain the speedup equation.

$$Speedup = \left(\frac{m}{\sqrt{\gamma_{k-1}}} + 1\right)^2 \tag{93}$$

8 Conclusion

This paper demonstrates the basic concept of how to optimally allocate divisible load and calculate speedup and solution time in tree networks where computing time is a nonlinear function of load size. Certainly the concept can be extended to other scheduling policies, higher order solutions and applications beyond sorting. Moreover it would be of interest to investigate the role of post-processing in such computing.

This work indicates that a (integer) χth order dependency of computation time on divisible problem size necessitates the solution of a χth order polynomial to solve for speedup. For the cases considered in this paper explicit solutions have been obtained. In the general case the optimal *allocation* of load in an N processor network necessitates the numerical solution of N nonlinear equations. Their format is similar to the used linear equation solution of linear divisible load models except for a power nonlinearity. It should be noted that numerical (arithmetic) problems may occur in solving such equations when χ is large.

This research result is extremely promising in providing a tractable means of assigning load to processors in an optimal manner for the important case of divisible loads with nonlinear computational complexity with its many applications.

References

[1] V. Bharadwaj, D. Ghose, and T.G. Robertazzi, "A new paradigm for load scheduling in distributed systems," *in special issue of Cluster Computing on Divisible Load Scheduling*, vol. 6, no. 1, pp. 7–18, Jan 2003, Kluwer Academic Publishers.

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

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

[4] J. T. Hung, H. J. Kim, and T. G. Robertazzi, "Scalable scheduling in parallel processors," *2002 Conference on Information Sciences and Systems*, March 2002, Princeton University.

[5] G. D. Barlas, "Collection aware optimum sequencing of operations and closed form solutions for the distribution of divisible load on arbitrary processor trees," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 5, pp. 429–441, May 1998.

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

[7] J. Blazewicz and M. Drozdowski, "Scheduling divisible jobs on hypercubes," *Parallel Computing*, vol. 21, no. 12, pp. 1945–1956, 1995.

[8] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert, "Bandwidth-centric allocation of independent tasks on heterogeneous platforms," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)*, June 2002.

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

[10] H. J. Kim, "A novel load distribution algorithm for divisible loads," *in special issue of Cluster Computing on Divisible Load Scheduling*, vol. 6, pp. 41–46, 2002.

[11] D. A. L. Piriyakumar and C. S. R. Murthy, "Distributed computation for a hypercube network of sensor-driven processors with communication delays including setup time," *IEEE Transactions on Systems, Man, and Cybernetics-PART A: Systems and Humans*, vol. 28, no. 2, pp. 245–251, March 1998.

[12] K. Li, "Parallel processing of divisible loads on partitionable static interconnection networks," *in special issue of Cluster Computing on Divisible Load Scheduling*, vol. 6, no. 1, pp. 47–56, January 2003, Kluwer Academic Publishers.

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

[14] J. Blazewicz, M. Drozdowski, F. Guinand, and D. Trystram, "Scheduling a divisible task in a 2-dimensional mesh," *Discrete Applied Mathematics*, p. 35, May 1999.

[15] W. Glazek, "A multistage load distribution strategy for three dimensional meshes," *in special issue of Cluster Computing on Divisible Load Scheduling*, vol. 6, no. 1, pp. 31–40, January 2003, Kluwer Academic Publishers.

[16] V. Bharadwaj, D. Ghose, and V. Mani, "Multi-installment load distribution in tree networks with delay," *IEEE Transaction on Aerospace and Electronic Systems*, vol. 31, no. 2, pp. 555–567, 1995.

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

[18] T. Robertazzi, "Ten reasons to use divisible load theory," *Computer*, vol. 36, no. 5, pp. 63–68, May 2003.

[19] M. Adler, Y. Gong, and A. L. Rosenberg, "Optimal sharing of bags of tasks in heterogeneous clusters," *SPAA'03*, June 2003, San Diego, California, USA.

[20] J. T. Hung and T. G. Robertazzi, "Divisible load cut through switching in tree networks," *(submitted for publication)*, 2003.

[21] S. H. Bokhari, "Assignment problems in parallel and distributed computing," *Kluwer Academic Publishers, Norwell, Mass*, 1987.

[22] C.S.R. Murthy and G. Manimaran, "Resource management in real time systems and networks," *MIT Press*, 2001.

[23] B.A. Shirazi, A.R. Hurson, and K.M. Kavi, "Scheduling and load balancing in distributed and parallel systems," *IEEE Computer Society Press, Los Alamitos, CA*, 96.

[24] K. A. Ross and C. R. B. Wright, *Discrete Mathematics*, Prentice Hall, 4 edition, 1999.