SCALABLE SCHEDULING FOR CLUSTERS AND GRIDS USING CUT THROUGH SWITCHING

J. T. Hung and T. G. Robertazzi

Abstract

A new scalable scheduling strategy using cut through switching is proposed in this paper. Recursive and closed form expressions for speedup are found in heterogeneous single level trees and in homogeneous multilevel trees, respectively. The ratio of speedup using cut through switching to that using store and forward switching is presented so as to illustrate the amount of improvement in speedup between these two different techniques.

Index Terms

Cut through switching, Divisible load scheduling, Scalability, Multilevel tree network, Simultaneous distribution

1 INTRODUCTION

The processing of massive amounts of data on distributed and parallel networks is becoming more and more common. Applications include grid computing, database applications, multimedia, and sensor network processing. Over the past 15 years, a number of researchers have mathematically modeled such processing using a divisible load scheduling model [1], which is useful for data parallel applications.

Divisible loads are ones that consist of data that can be arbitrarily partitioned among a number of processors interconnected through some network. Divisible load modeling usually assumes no precedence relations amongst the data. Due to the linearity of the divisible model, optimal scheduling strategies under a variety of environments can be devised.

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 seamless integrated manner. Moreover, it is tractable with its linearity assumption. It has been used to accurately and directly model such features as specific network topologies and scheduling policies [1], [2], [3], [4], [5], [6], [7], [8], [9], computation versus communication load intensity [1], [2], time varying inputs [10], multiple job submission [1], [11], [12], [13], and numerous applications. However, researchers in this field have noted an saturation limit. If speedup (or solution time) is considered as a function of the number of processors, an asymptotic constant is reached as the number of processors is increased. Beyond a certain point, adding processors results in minimal performance improvement.

For the first interconnection topology considered in the literature, the linear daisy chain [2], the saturation limit is usually explained by noting that, if load originates at a processor at a boundary of the chain, data must be transmitted and retransmitted $i-1$ times from processor to processor before it arrives at the ith processor (assuming nodes with store and forward transmission). However, for subsequent interconnection topologies considered (e.g. bus, single level tree, hypercube), the reason for this lack of scalability has been less obvious.

The reason why the saturation occurs in a single level tree is because of the assumption that a node distributes load sequentially to one of its children at a time. This is true for both single and multi-installment scheduling strategies discussed to date [1], [5]. In a single level tree (star topology), if a processor can distribute load to all of its children simultaneously, the speedup is a linear function of the number of processors [14].

How to implement simultaneous (concurrent) distribution is indicated in [14] as follows:

> *How might one implement the strategy that a processor distributes load concurrently to all its neighbors? A direct method, similar to what is done in packet switches, is to envision that a processor has a CPU and an output buffer for each output link. Scalability can be achieved as long as the CPU can effectively distribute load to all of its output buffers concurrently.*

In a multilevel tree, even though the hardware can support the mechanism that the processors distribute load to their children simultaneously, saturation can also occur if scheduling policies do not adopt the cut through switching policy from upper level to lower level and the simultaneous distribution policy in the same level. A fat tree network architecture was introduced as one means for hardware to implement simultaneous distribution so as to avoid the saturation problem [14]. The actual problem

Thomas G. Robertazzi, Cosine Laboratory, Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. 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. E-mail: trent@ece.sunysb.edu

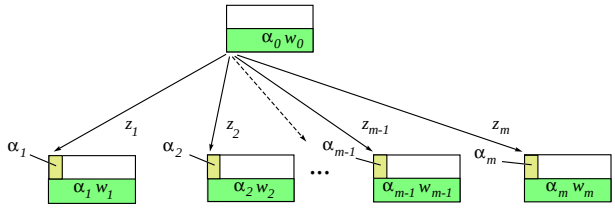

Fig. 1. Structure of a heterogeneous single level tree with simultaneous distribution, simultaneous start.

in multilevel trees is the nature of store and forward switching, which is used in [14], down a path in such a tree. With an efficient scheduling policy (such as cut through switching with simultaneous distribution) it is ultimately hardware that limits the performance. This is in the sense that single level tree scheduling is scalable as long as outputs of the root are continually loaded.

This paper considers the use of cut through switching to overcome this problem. In cut through switching, load proceeding down a multilevel tree path need not be completely received by a node before it can be forwarded to the node's descendants. Rather, a node can simultaneously receive load from its parent node and transmit (from the portion received so far) to its children at the same time. Intuitively one can see that this should lead to a performance improvement but we seek here to quantify that performance.

Note that simultaneous load scheduling was proposed by Piriyakumar and Murthy in [15]. While they stated this should lead to improve performance, the scalable nature of simultaneous scheduling was not known until [14]. Previous related work on scalability issues for parallel processing includes an experimental study of several real time load balancing schemes [16] and an experimental study of scalable scheduling for function parallelism on distributed memory multiprocessors [17]. It has been known on an intuitive basis that network elements should be kept constantly busy for good performance [18].

This paper presents the types of notation and analytic background in Section 2. In Section 3 the speedup formulas for processors are derived for a single level and a multilevel tree, respectively. We also compare the speedup using cut through switching (with simultaneous start) with that using store and forward switching (with simultaneous start) [19]. The conclusion is stated in Section 4.

2 Model and Notation

2.1 Model and Notation for Single Level Tree

To evaluate a homogeneous multilevel tree, we must analyze a single level tree first. Here we present a heterogeneous single level tree, (see Figure 1), with intelligent root. All the children processors are connected to the root (parent) processor via communication links. An intelligent root can process a fraction of the load as well as distribute the remaining load to its children processors at the same time. Timing diagrams within each node of Figure 1 show communication above the horizontal time axis and computation below it. In this paper we assume that a node begins to process its load as soon as load is received by the nodes as proposed by Kim [20]. This is simultaneous start.

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 on the ith processor.
w_{eq}: The inverse computing speed on an equivalent node collapsed from a single level tree.
z_i: The inverse link speed on the ith link.
T_{cp}: Computing intensity constant. The entire load can be processed in $w_i T_{cp}$ seconds on the ith processor.
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 collapsed from a single level tree composed of one root node and m children nodes. Here $T_{f,m}$ is equal to $w_{eq} T_{cp}$.
$T_{f,0}$: The finish time for the entire divisible load solved on the root processor, (i.e. a tree without any children nodes but the root node). Here $T_{f,0}$ is equal to $1 \times w_0 T_{cp}$, that is $w_0 T_{cp}$.

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

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

Definition 2: Speedup, the ratio of finish time on one processor (i.e. the root node) to that on an equivalent node collapsed from a single level tree. It is thus a measure of parallel processing advantage. 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:

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

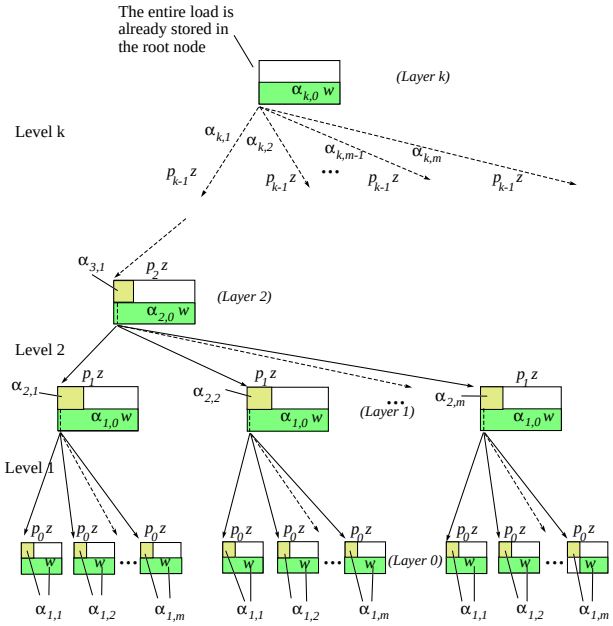

Fig. 2. Structure of a homogeneous multilevel fat tree using cut through switching, simultaneous distribution, and simultaneous start.

2.2 Model and Notation for Multilevel Tree

A heterogeneous multilevel tree network is too complex for a closed form speedup solution. Therefore, in this paper a homogeneous multilevel tree network is evaluated. A homogeneous multilevel tree network where root processors are equipped with a front-end processor for off-loading communications is considered. Root nodes, called intelligent roots, process a fraction of the load as well as distribute the remaining load to their children processors at the same time (see Figure 2). Note that each child processor starts computing and transmitting immediately as soon as it receives its assigned fraction of load and continues without any interruption until all of its assigned load fraction has been processed. Under simultaneous distribution load is distributed from a node to its children simultaneously, as opposed to sequentially. This is the operation of "cut through switching with simultaneous distribution" for computation and communication.

An equivalent subtree can be obtained by collapsing the lower level subtrees into an equivalent node [21]. Therefore, an equivalent network for a level j subtree in a multilevel tree can be derived and is illustrated in Figure 3.

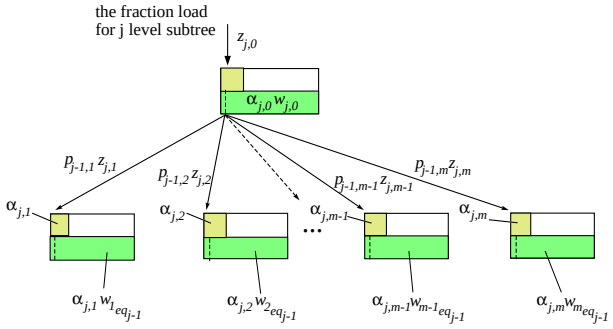

Fig. 3. Structure of j-level subtree in a multilevel tree using cut through switching, simultaneous distribution, and simultaneous start.

The notation for a multilevel homogeneous fat tree is denoted as follows.

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

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

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

$T^{h,k}_{f,m}$: 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 Figure 3). The value of the multiplier $p_{j-1,i}$ is defined as the inverse of the total number of children processor descendants at and below layer $j-1$ 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, $p_{j-1} = p_{j-1,i}$ $(i = 1, 2, \ldots, m)$. Hence:

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

With this natural choice of p_{j-1}, the transmission capacity between parent nodes and children nodes is $1/(p_{j-1}z)$, which is larger than the capacity of bottommost level links by $1/p_{j-1}$. This implicitly indicates that each node within an equivalent subtree from layer $j-1$ down to layer 0 has an equivalent capacity of $1/z$ in the channel at level j.

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 collapsed from 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:

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

We make three major assumptions. First, the computing and communication loads are divisible (i.e. perfectly divisible with no precedence constraints [1]). Second, transmission and computation time are proportional (linear) to the size of the problem transmitted or computed. Finally, each node transmits load simultaneously to its children.

3 Cut Through Switching with Simultaneous Distribution and Simultaneous Start

This section examines scalable scheduling with two improved features. One feature is the use of cut through switching, rather than store and forward switching [14], for load distribution from level to level in a tree network. The second feature is to allow a child node receiving load from its parent node to begin computing as soon as load starts to arrive. This type of timing, referred to here as simultaneous start, was first proposed by Kim [20].

An ideal form of cut through switching would provide sufficient virtual circuits (one from the root to each node) so that all nodes in the multilevel tree receive load simultaneously with effective root to individual node data rate of $1/z$ (or some variation for a heterogeneous network). Neglecting propagation delays, this creates a logical single level tree (star) network. However, because there is more traffic as one approaches the root, a fat tree like assignment of link capacity is needed.

The form of cut through switching used in this study is to have the multilevel tree root start to transmit simultaneously to all its children. After a node receives its own load, load for the node's children will be relayed (in virtual cut through) through the node to its children. Each node distributes load to its children in a similar manner. Each node receiving load commences processing on its fraction as it starts to receive it. Once a node has received its own load it continues processing and relays load to its children and descendants.

Finally, in this work we proceed by aggregating single level subtrees into equivalent processors, starting from the bottom to the tree and working upwards [1].

The tree's bottommost single level subtrees are at level 1, the tree's topmost (including the root) subtree is at level k. A little thought will show that the topmost single level subtree can have all processors commence computation simultaneously at time zero (root node with data storage case).

In the lower level sub-trees there is a delay between the time that a subtree's root commences computation and reception of its load and the time it finishes reception of its own load and can begin distributing load to its children, so they can start processing (root node without data storage case).

In this paper the single level tree root node with data storage and root node without data storage cases are first examined. Then these results are applied to a multi-level tree. Again, root node without data storage scheduling is applied to lower levels, $1, 2, \ldots, k-1$, and root node with data storage scheduling is applied to topmost level k.

3.1 Processors with Simultaneous Start: Single Level Tree

3.1.1 Single Level Tree: Root Node with Data Storage: Consider a single level tree network with an intelligent root, $m+1$ processors, and m links. 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 into $m+1$ fractions, keeps its own fraction α_0, and distributes the other fractions α_1, α_2, ..., α_m to the children processors respectively and simultaneously.

Each processor begins computing immediately as soon as load starts to arrive and continues without any interruption until all of its assigned load fraction has been processed. In order to minimize the processing finish time, all of the utilized processors

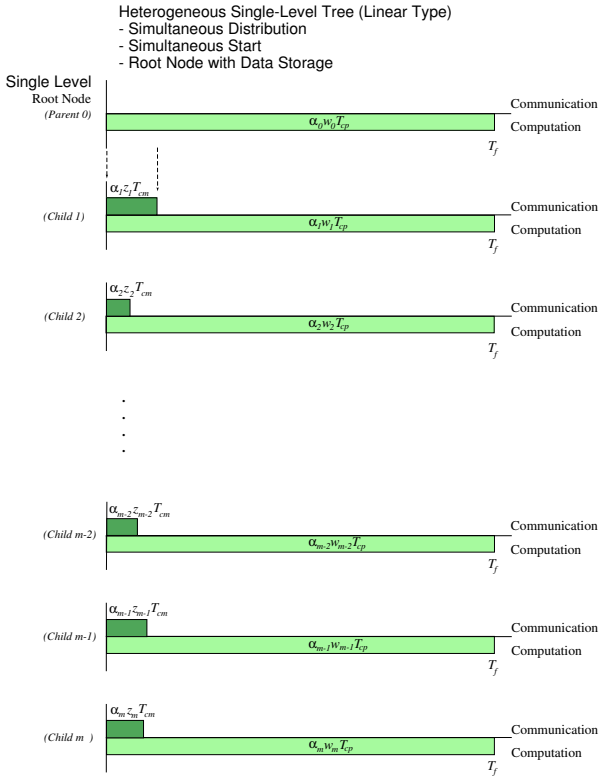

Fig. 4. Timing diagram of single level tree with simultaneous distribution, simultaneous start, and root node with data storage.

in the network must finish computing at the same time [1]. The process of load distribution can be represented by Gantt chart-like timing diagrams, as illustrated in Figure 4. Note that this is a completely deterministic model. The fundamental recursive equations of the system can be formulated as follows:

$$\alpha_0 w_0 T_{cp} = \alpha_1 w_1 T_{cp} \tag{6}$$

$$\alpha_{i-1} w_{i-1} T_{cp} = \alpha_i w_i T_{cp} \qquad i = 2, 3, \ldots, m \tag{7}$$

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

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{8}$$

This gives $m+1$ linear equations with $m+1$ unknowns.
Now from (6):

$$\alpha_0 = \frac{w_1}{w_0}\alpha_1 = \frac{1}{k_1}\alpha_1 \qquad \text{where} \quad k_1 = w_0/w_1. \tag{9}$$

Then from (7),

$$\alpha_i = \frac{w_{i-1}T_{cp}}{w_i T_{cp}}\alpha_{i-1} = q_i \alpha_{i-1} \quad \text{where} \quad q_i = w_{i-1}/w_i. \tag{10}$$

$$= (\prod_{l=2}^{i} q_l) \times \alpha_1 \qquad i = 2, 3, \ldots, m \tag{11}$$

Then the normalization equation (8) becomes:

$$\left[\frac{1}{k_1} + 1 + \sum_{i=2}^{m}(\prod_{l=2}^{i} q_l)\right]\alpha_1 = 1$$

$$\alpha_1 = \frac{1}{\frac{1}{k_1} + 1 + \sum_{i=2}^{m}(\prod_{l=2}^{i} q_l)} \tag{12}$$

The finish time is:

$$T_{f,m} = \alpha_0 w_0 T_{cp} = \frac{1}{k_1}\alpha_1 w_0 T_{cp} \tag{13}$$

$$= \frac{1}{1 + k_1\left[1 + \sum_{i=2}^{m}(\prod_{l=2}^{i} q_l)\right]} w_0 T_{cp} \tag{14}$$

The single level tree can be collapsed into a single equivalent node, and the equivalent computation speed w_{eq} is:

$$w_{eq} T_{cp} = T_{f,m} = \frac{1}{1 + k_1\left[1 + \sum_{i=2}^{m}(\prod_{l=2}^{i} q_l)\right]} w_0 T_{cp}$$

According to Definition 1 in Section 2:

$$\gamma_{eq} = \frac{w_{eq}}{w_0} = \frac{1}{1 + k_1\left[1 + \sum_{i=2}^{m}\frac{w_1}{w_i}\right]} \tag{15}$$

Since $T_{f,0} = \alpha_0 w_0 T_{cp}$ and $\alpha_0 = 1$: the speedup is derived as follows:

$$Speedup = \frac{T_{f,0}}{T_{f,m}} = \frac{1}{\gamma_{eq}} = 1 + k_1\left[1 + \sum_{i=2}^{m}\frac{w_1}{w_i}\right] \tag{16}$$

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 (i.e. $w_i = w$ and $z_i = z$ for $i = 1, 2, \ldots, m$). Note that w_0 can be different from w_i, and then $k_1 = w_0/w$ and $q_i = 1$ ($i = 2, 3, \ldots, m$). Consequently, the speedup can be a linear function of the number of children processors as follows.

$$Speedup = 1 + k_1\left[1 + \sum_{i=2}^{m}\frac{w_1}{w_i}\right] = 1 + m\frac{w_0}{w} \tag{17}$$

3.1.2 Single Level Tree: Root Node without Data Storage: Here we assume that the children nodes in the subtree begin computing when the root node finishes receiving its load. It is assumed that communication speed is fast enough on links that no node "starves" for load. The Gantt chart-like timing diagram of this process of load distribution can be illustrated in Figure 5.

The fundamental recursive equations of the system can be formulated as follows:

$$\alpha_0 w_0 T_{cp} = \alpha_1 w_1 T_{cp} + \alpha_0 z_0 T_{cm} \tag{18}$$

$$\alpha_{i-1} w_{i-1} T_{cp} = \alpha_i w_i T_{cp} \qquad i = 2, 3, \ldots, m \tag{19}$$

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

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{20}$$

This gives $m + 1$ linear equations with $m + 1$ unknowns. Now from (18),

$$\alpha_0 = \frac{w_1 T_{cp}}{w_0 T_{cp} - z_0 T_{cm}}\alpha_1 = \frac{1}{k_1}\alpha_1 \tag{21}$$

Here:

$$k_1 = \frac{w_0 T_{cp} - z_0 T_{cm}}{w_1 T_{cp}} \tag{22}$$

Here we assume that $w_0 T_{cp} > z_0 T_{cm}$. That is, communication speed is faster than computation speed. Following the derivations in section 3.1.1, we can obtain the following results:

$$T_{f,m} = \alpha_0 w_0 T_{cp} = \frac{1}{k_1}\alpha_1 w_0 T_{cp} \tag{23}$$

$$= \frac{1}{1 + k_1\left[1 + \sum_{i=2}^{m}(\prod_{l=2}^{i} q_l)\right]} w_0 T_{cp} \tag{24}$$

$$\gamma_{eq} = \frac{w_{eq}}{w_0} = \frac{1}{1 + k_1\left[1 + \sum_{i=2}^{m}\frac{w_1}{w_i}\right]} \tag{25}$$

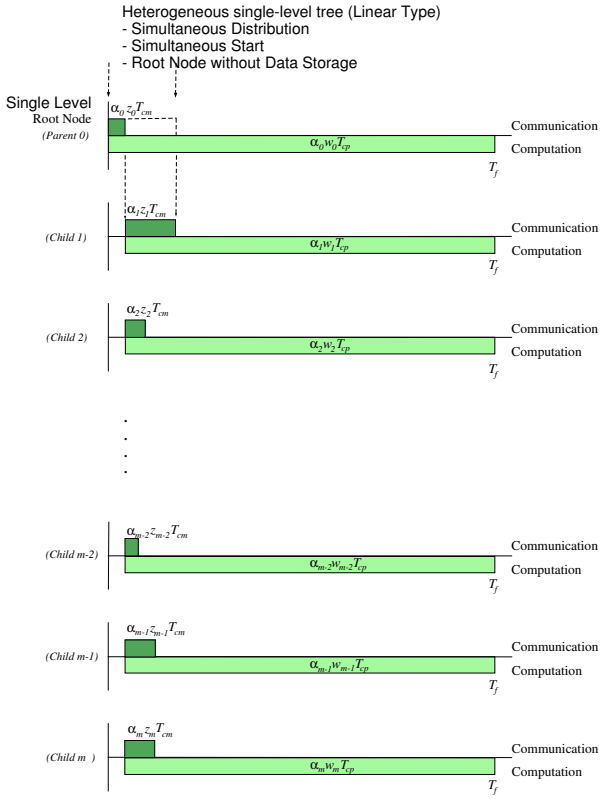

Fig. 5. Timing diagram of single level tree with simultaneous distribution, simultaneous start, and root node without data storage.

$$Speedup = \frac{T_{f,0}}{T_{f,m}} = \frac{1}{\gamma_{eq}} = 1 + k_1 \left[1 + \sum_{i=2}^{m} \frac{w_1}{w_i} \right] \tag{26}$$

As a special case, (i.e. $w_i = w$ and $z_i = z$ for $i = 1, 2, \ldots, m$):

$$Speedup = 1 + m(w_0/w - z_0\sigma/z) \tag{27}$$

Here $\sigma = zT_{cm}/wT_{cp}$.

3.2 Processors with Simultaneous Start: Homogeneous Multiple Level Tree Analysis

The process of load distribution for the multilevel fat tree network using cut through strategy for computing and communicating can be represented by Gantt chart-like timing diagram in Figure 6.

The dashed blocks at the communication parts denote the periods when data only passes through a node with cut through switching and the solid blocks at the communication parts denote the periods when the indicated node receives the data assigned for computation.

3.2.1 Level j Subtree: Single Level Tree Root Node without Data Storage: The timing diagram of a single equivalent jth level tree is analogous to Figure 5. However, the following notation is defined as $\alpha_i = \alpha_{j,i}$ $(i = 0, 1, 2, \ldots, m)$; $z_0 = p_j z$; $z_i = p_{j-1} z$ $(i = 1, 2, \ldots, m)$; $w_0 = w$; and $w_i = w_{eq_{j-1}}$ $(i = 1, 2, \ldots, m)$. Consequently, the fundamental recursive equations of the jth level subtree network are derived as follows:

$$\alpha_{j,0} w T_{cp} = \alpha_{j,1} w_{eq_{j-1}} T_{cp} + \alpha_{j,0} p_j z T_{cm} \tag{28}$$

$$\alpha_{j,i-1} w_{eq_{j-1}} T_{cp} = \alpha_{j,i} w_{eq_{j-1}} T_{cp} \qquad i = 2, 3, \ldots, m \tag{29}$$

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

$$\alpha_{j,0} + \alpha_{j,1} + \alpha_{j,2} + \cdots + \alpha_{j,m} = 1 \tag{30}$$

This gives $m + 1$ linear equations with $m + 1$ unknowns. Then from (28):

$$\alpha_{j,0} = \frac{w_{eq_{j-1}} T_{cp}}{w T_{cp} - p_j z T_{cm}} \alpha_{j,1} = \frac{1}{k_{eq_{j-1}}} \alpha_{j,1} \tag{31}$$

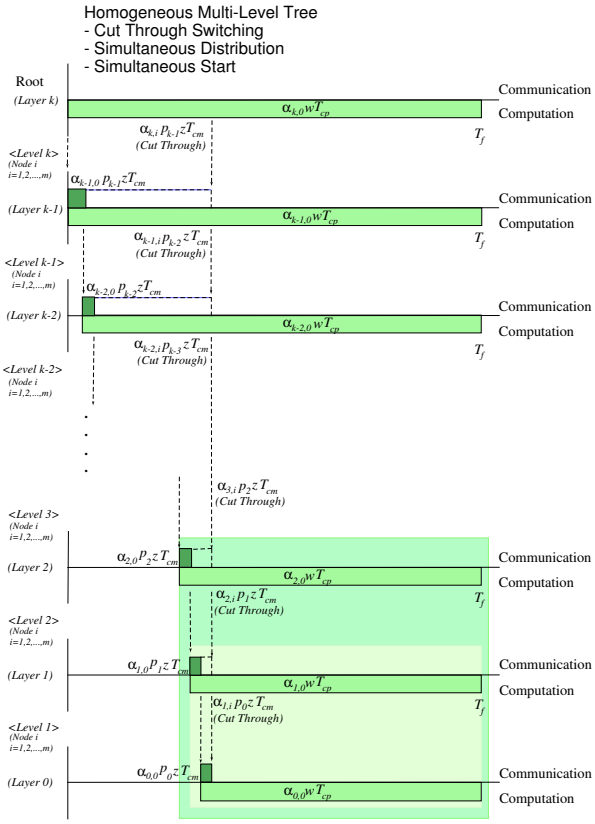

Fig. 6. Timing diagram for scalable scheduling using cut through switching, simultaneous distribution, and simultaneous start in a multilevel fat tree.

According to Definition 4 in Section 2 and $\sigma = zT_{cm}/wT_{cp}$, $k_{eq_{j-1}}$ becomes:

$$\begin{aligned} k_{eq_{j-1}} &= \frac{wT_{cp} - p_j zT_{cm}}{w_{eq_{j-1}} T_{cp}} = \frac{w}{w_{eq_{j-1}}} - \frac{wp_j zT_{cm}}{w_{eq_{j-1}} wT_{cp}} \\ &= \frac{w}{w_{eq_{j-1}}}(1 - p_j\sigma) = \frac{1}{\gamma_{j-1}}(1 - p_j\sigma) \end{aligned} \tag{32}$$

Now from (29):

$$\begin{aligned} \alpha_{j,i} &= \frac{w_{eq_{j-1}} T_{cp}}{w_{eq_{j-1}} T_{cp}} \alpha_{j,i-1} = q_{eq_{j-1}} \alpha_{j,i-1} = \left[\prod_{l=2}^{i} q_{eq_j - 1} \right] \alpha_{j,1} \\ &= (q_{eq_{j-1}})^{i-1} \alpha_{j,1} \qquad i = 2, 3, \ldots, m \end{aligned} \tag{33}$$

Here, as the network is homogeneous (see (33)), $q_{eq_{j-1}} = 1$. The normalization equation (30) becomes:

$$\frac{1}{k_{eq_{j-1}}} \alpha_{j,1} + \alpha_{j,1} + \sum_{i=2}^{m} \alpha_{j,i} = 1 \tag{34}$$

$$\alpha_{j,1} = \frac{1}{\frac{1}{k_{eq_{j-1}}} + 1 + \sum_{i=2}^{m} \left[\prod_{l=2}^{i} q_{eq_{j-1}} \right]} \tag{35}$$

Therefore, the equivalent finish time is as follows:

$$T_{f,m}^{h,j} = \alpha_{j,0} wT_{cp} = \frac{1}{k_{eq_{j-1}}} \alpha_{j,1} wT_{cp} \tag{36}$$

$$= \frac{1}{1 + k_{eq_{j-1}} \left\{ 1 + \sum_{i=2}^{m} \left[\prod_{l=2}^{i} q_{eq_{j-1}} \right] \right\}} wT_{cp} \tag{37}$$

Since $w_{eq_j} T_{cp} = T^{h,j}_{f,m}$ and $k_{eq_{j-1}} = (1 - p_j\sigma)/\gamma_{j-1}$, we obtain

$$w_{eq_j} = \frac{1}{1 + k_{eq_{j-1}} \left\{1 + \sum_{i=2}^{m} \left[\prod_{l=2}^{i} q_{eq_{j-1}}\right]\right\}} w$$

$$\begin{aligned}\gamma_j = \frac{w_{eq_j}}{w} &= \frac{1}{1 + k_{eq_{j-1}} \left\{1 + \sum_{i=2}^{m} \left[\prod_{l=2}^{i} q_{eq_{j-1}}\right]\right\}} \\ &= \frac{1}{1 + \frac{m}{\gamma_{j-1}}(1 - p_j\sigma)} \end{aligned} \tag{38}$$

From (38), the closed forms of speedup for fat tree and non-fat tree networks are obtained as follows.

1) *Homogeneous multilevel fat tree*:
Since the computation capability of each node is w, w_{eq_0} is equal to w. Therefore, the initial value of γ, (which is γ_0), is equal to 1 (because $\gamma_0 = w_{eq_0}/w = w/w$). Furthermore, γ_j can be derived as follows:

$$\gamma_1 = \frac{1}{1 + \frac{m}{\gamma_0}(1 - p_1\sigma)} = \frac{1}{1 + m(1 - p_1\sigma)} \tag{39}$$

$$\begin{aligned}\gamma_2 &= \frac{1}{1 + \frac{m}{\gamma_1}(1 - p_2\sigma)} \\ &= \frac{1}{1 + m(1 - p_2\sigma) + m^2(1 - p_2\sigma)(1 - p_1\sigma)} \end{aligned}$$

$$\gamma_3 = \frac{1}{R(\sigma, m)} \tag{40}$$

$$\vdots$$

$$\gamma_j = \frac{1}{1 + \sum_{i=1}^{j} \left[\prod_{l=0}^{i-1} m(1 - p_{j-l}\sigma)\right]} \tag{41}$$

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

$$\begin{aligned}R(\sigma, m) = 1 &+ m(1 - p_3\sigma) + m^2(1 - p_3\sigma)(1 - p_2\sigma) \\ &+ m^3(1 - p_3\sigma)(1 - p_2\sigma)(1 - p_1\sigma) \end{aligned} \tag{42}$$

The equivalent speedup of level j subtree is described as the following equation:

$$Speedup = \frac{1}{\gamma_j} = 1 + \sum_{i=1}^{j} \left[\prod_{l=0}^{i-1} m(1 - p_{j-l}\sigma)\right] \tag{43}$$

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

2) *Homogeneous multilevel non-fat tree* (all the bandwidth of each transmission link is the same, $p_j = 1$):
In (41), $p_j = 1$. Hence, the closed form solution of γ_j is:

$$\gamma_j = \frac{1}{\sum_{i=0}^{j} [m(1 - \sigma)]^i} \qquad j = 1, 2, \ldots, k-1 \tag{44}$$

The equivalent speedup of a subtree, level j, is obtained as follows:

$$Speedup = \frac{1}{\gamma_j} = \sum_{i=0}^{j} [m(1 - \sigma)]^i = \frac{1 - [m(1 - \sigma)]^{j+1}}{1 - m(1 - \sigma)} \tag{45}$$

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

3.2.2 Level k Subtree: Root Node with Data Storage: The timing diagram of the top equivalent single level tree, level k, is analogous to Figure 4. However, the following notation is defined as $\alpha_i = \alpha_{k,i}$ $(i = 0, 1, 2, \ldots, m)$; $z_i = p_{k-1}z$ $(i = 1, 2, \ldots, m)$; $w_0 = w$; and $w_i = w_{eq_{k-1}}$ $(i = 1, 2, \ldots, m)$. Consequently, the fundamental recursive equations of the kth-level subtree are derived as follows:

$$\alpha_{k,0} w T_{cp} = \alpha_{k,1} w_{eq_{k-1}} T_{cp} \tag{46}$$

$$\alpha_{k,i-1} w_{eq_{k-1}} T_{cp} = \alpha_{k,i} w_{eq_{k-1}} T_{cp} \qquad i = 2, 3, \ldots, m \tag{47}$$

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

$$\alpha_{k,0} + \alpha_{k,1} + \alpha_{k,2} + \cdots + \alpha_{k,m} = 1 \tag{48}$$

Applying the derivations in section 3.2.1, one obtained the following results. So from (46):

$$\alpha_{k,0} = \frac{w_{eq_{k-1}} T_{cp}}{w T_{cp}} \alpha_{k,1} = \frac{1}{k_{eq_{k-1}}} \alpha_{k,1} \tag{49}$$

where

$$k_{eq_{k-1}} = \frac{w T_{cp}}{w_{eq_{j-1}} T_{cp}} = \frac{w}{w_{eq_{j-1}}} = \frac{1}{\gamma_{k-1}} \tag{50}$$

$$T^{h,k}_{f,m} = \alpha_{k,0} w T_{cp} = \frac{1}{k_{eq_{k-1}}} \alpha_{k,1} w T_{cp} \tag{51}$$

$$= \frac{1}{1 + m k_{eq_{k-1}}} w T_{cp} \tag{52}$$

$$\gamma_k = \frac{w_{eq_k}}{w} = \frac{1}{1 + m k_{eq_{k-1}}} = \frac{1}{1 + \frac{m}{\gamma_{k-1}}} \tag{53}$$

Therefore, we can summarize the speedup of level k. If $k = 1$, then $\gamma_0 = 1$ and $\gamma_1 = \gamma_0/(m + \gamma_0)$. If $k \geq 2$, the closed form solution for a k level fat tree is shown as follows.

1) *Homogeneous multilevel fat tree*:

$$Speedup = \frac{1}{\gamma_k}$$
$$= 1 + m \left\{ 1 + \sum_{i=1}^{k-1} \left[\prod_{l=0}^{i-1} m(1 - p_{(k-1)-l}\sigma) \right] \right\}$$

2) *Homogeneous multilevel non-fat tree*, (all the bandwidth of each transmission link is the same, $p_j = 1$):

$$Speedup = \frac{1}{\gamma_k} = 1 + m \sum_{i=0}^{k-1} [m(1 - \sigma)]^i$$

3.2.3 Speedup Comparison Between Cut Through Switch and Store and Forward Switch: In [19] the recursive formulae for a homogeneous multilevel fat tree with simultaneous start using store and forward switching are $\gamma_0 = 1$; $\gamma_j = (p_{j-1}\sigma(\gamma_{j-1} + m) + \gamma_{j-1})/(\gamma_{j-1} + m)$ $(j = 1, 2, \ldots, k-1)$; and $\gamma_k = \gamma_{k-1}/(m + \gamma_{k-1})$. We can thus solve these formulae to obtain the ration of speedup using cut-through switching to that using store and forward switching (see Figure 7, $\sigma = 0.1$). As Figure 7

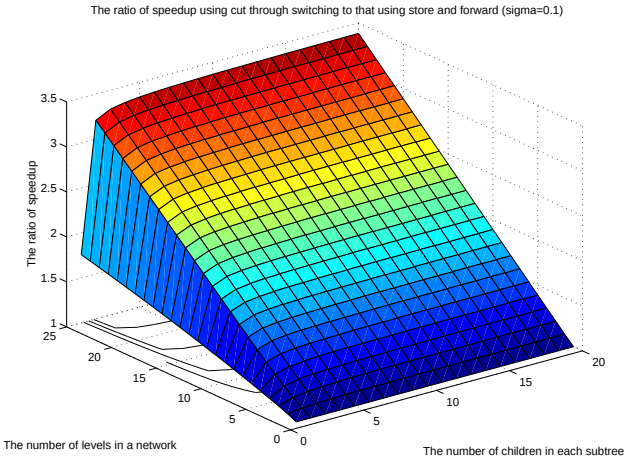

Fig. 7. The ratio of speedup of cut-through switch to that of store and forward switch ($\sigma = 0.1$)

shows, speedup using cut through switching is faster than that using store and forward switching.

4 CONCLUSIONS

Tree networks are of direct relevance to cluster and grid interconnection. Spanning trees are an efficient and commonly used interconnection topology in large scale networks/grids for distributing load while minimizing the number (and inherent cost) of links used. In clusters, single level trees (i.e. stars) are a natural topology for small scale interconnection and multilevel trees are natural for larger system interconnection.

The performance analysis of a heterogeneous single level tree and a homogeneous multilevel tree using cut through switching was derived in this paper. The speedup of scheduling using cut through switching outperforms than that of scheduling using store and forward switching. A comparison in the performance between these two method becomes possible by solving the recursive equations derived in this paper.

REFERENCES

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

[2] Y. C. Cheng and T. G. Robertazzi, Distributed computation with communication delays, *IEEE Transactions on Aerospace and Electronic Systems*, 24(6), 1988, 700-712.

[3] 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*, 9(5), 1998, 429-441.

[4] S. Bataineh and T. G. Robertazzi, Bus oriented load sharing for a network of sensor driven processors, *IEEE Transactions on Systems, Man and Cybernetics*, 21(5) 1991, 1202-1205.

[5] V. Bharadwaj, D. Ghose, and V. Mani, Multi-installment load distribution in tree networks with delay, *IEEE Transaction on Aerospace and Electronic Systems*, 31(2), 0 1995, 555-567.

[6] V. Bharadwaj, D. Ghose, and V. Mani, An efficient load distribution strategy for a distributed linear network of processors with communication delays, *Computers and Mathematics with Applications*, 29(9), 1995, 95-112.

[7] J. Blazewicz and M. Drozdowski, Scheduling divisible jobs on hypercubes, *Parallel Computing*, 21(12), 1995, 1945-1956.

[8] J. Blazewicz and M. Drozdowski, The performance limits of a two dimensional network of load sharing processors, *Foundations of Computing and Decision Sciences*, 21(1), 1996, 3-15.

[9] J. Blazewicz and M. Drozdowski, Distributed processing of divisible jobs with communication start-up costs, *Discrete Applied Mathematics*, 76(1-3), 1997, 21-41.

[10] J. Sohn and T. G. Robertazzi, Optimal time varying load sharing for divisible loads, *IEEE Transactions on Aerospace and Electronic Systems*, 34(3), 1998, 907-924.

[11] V. Bharadwaj and G. Barlas, Efficient scheduling strategies for processing multiple divisible loads on bus networks, *Journal of Parallel and Distributed Computing*, 62, 2002, 132-151.

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

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

[14] J. T. Hung, H. J. Kim, and T. G. Robertazzi, Scalable scheduling in parallel processors, *Proc. 2002 Conference on Information Sciences and Systems*, 2002.

[15] 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*, 28(2), 1998, 245-251.

[16] V. Kumar, A. Y. Grama, and N. R. Vempaty, Scalable load balancing techniques for parallel computers, *Journal of Parallel and Distributed Computing*, 22, 1994, 60-79.

[17] S. Pande, D. P. Agrawal, and J. Mauney, A scalable scheduling scheme for functional parallelism on distributed memory multiprocessor systems, *IEEE Transactions on Parallel and Distributed Systems*, 6, 1995, 388-399.

[18] D. E. Culler and J. P. Singh, *Parallel Computer Architecture*, (San Francisco: Morgan Kaufmann, 1999).

[19] J. T. Hung, *Scalable Scheduling in Parallel, Distributed, and Grid Systems*, doctoral diss., Stony Brook University, Stony Brook, NY, 2003.

[20] H. J. Kim, A novel load distribution algorithm for divisible loads, *Special Issue of Cluster Computing on Divisible Load Scheduling*, 6(1), 2002, 41-46.

[21] S. Bataineh, T. Y. Hsiung, and T. G. Robertazzi, Closed form solutions for bus and tree networks of processors load sharing a divisible job, *IEEE Transactions on Computers*, 43(10), 1994, 1184-1196.

PLACE PHOTO HERE

Jui Tsun Hung received the M.S. and Ph.D. degrees in Electrical Engineering from State University of New York at Stony Brook in 2001 and 2003, respectively.

He also received the B.S.M.E. degree from National Sun Yat-Sen University and M.S.M.E. degree from Chuang Yuan Christian University in Taiwan.

His current interests are focused on analyzing the performance of computing networks.

PLACE
PHOTO
HERE

Thomas G. Robertazzi received the Ph.D from Princeton University in 1981.

He is a Professor at Stony Brook University in the Dept. of Electrical and Computer Engineering. His research interests include grid computing, scheduling, networking and performance evaluation. He has authored, co-authored and edited four books in these areas.