Scalable Scheduling in Parallel Processors

Jui Tsun Hung[1] Hyoung Joong Kim[2] Thomas G. Robertazzi[3]

Abstract — **A study of scalable data intensive scheduling involving load distribution on multi-level fat tree networks is presented.**

It is shown that the speedup of processing divisible loads concurrently on a homogeneous single level tree increases linearly as the number of processors is increased. This behavior is markedly different from that of past research on sequential scheduling where speedup saturates beyond a certain number of processors.

It is demonstrated that the ultimate limiting factor for speedup is due to hardware, not to the scheduling strategy on a multi-level fat tree.

I. Introduction

The processing of massive amounts of data on distributed and parallel networks is becoming more and more common. Applications include large scientific experiments, database applications, image processing, and sensor data processing. Over the past dozen years, a number of researchers have mathematically modeled such processing using a divisible load scheduling model [1] ,that is useful for data parallelism 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 have been devised.

The majority of the divisible load scheduling literature has appeared in computer engineering periodicals. However, divisible load modeling should be of interest to the networking community as

1. it models, both computation and network communication in a completely seamless integrated manner, and
2. it is tractable with its linearity assumption.

It has been used to accurately and directly model such features as specific network topologies [1-10], computation versus communication load intensity [1][2], time varying inputs [10], multiple job submission [1], and monetary cost optimization [11][12].

However, researchers in this field have noted an important performance 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. In other words, the scheduling strategies considered to date have not been scalable.

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 a node 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.

In this paper, it is demonstrated that the reason why the saturation occurs 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]. It is shown here that in a single level tree (star topology), if a processor can distribute load to all of its children concurrently, the speedup is a linear function of the number of processors. The only scalability limitations are a proportionality constant which depends on system parameters and the ability of a processor to distribute loads concurrently to all of its outgoing links.

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. Naturally, as the number of neighbors of a processor node is increased, a point will be reached while the CPU can not load the buffers as fast as those buffers are being emptied. However, for a given CPU, this architecture allows a better use of the time shared CPU up to a large number of children nodes than the earlier assumption that a CPU distributes load sequentially and completely to one output buffer and link at a time. Certainly, one can also envision high performance architectures such as multiple CPUs within the same node. Through a consideration of concurrent load distribution by a node to its neighbors, one can see that the ultimate limitation is on hardware, not scheduling, as it should be.

Note that previous related work on scalability issues for parallel processing includes an experimental study of several real time load balancing schemes [13] and an experimental study of scalable scheduling for function parallelism on distributed memory multiprocessors [14]. It has been known on an intuitive basis that network elements should be kept constantly busy for good performance [15]. This research mathematically quantifies that. This paper begins with the development of some notation and analytic background in section II. The parent nodes do both load distribution and data computation in section III. The conclusion appears in section IV.

II. Model, Notation and Load Distribution

In this paper, a homogeneous multi-level fat tree network where root processors are equipped with a front-end proces-

[1]Jui Tsun Hung, Department of Electrical and Computer Engineering, University at Stony Brook, E-mail: trent@ece.sunysb.edu

[2]Hyoung Joong Kim, Department of Control and Instrumentation Engineering, Kangwon National University, E-mail: khj@cc.kangwon.ac.kr

[3]Thomas G. Robertazzi, a Senior Member, IEEE, Associate Professor of Department of Electrical and Computer Engineering, University at Stony Brook, Stony Brook, NY 11794. Phone (631) 632-8400, Fax (631) 632-8494, E-mail: tom@ece.sunysb.edu

sor 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, (see Fig. 1).

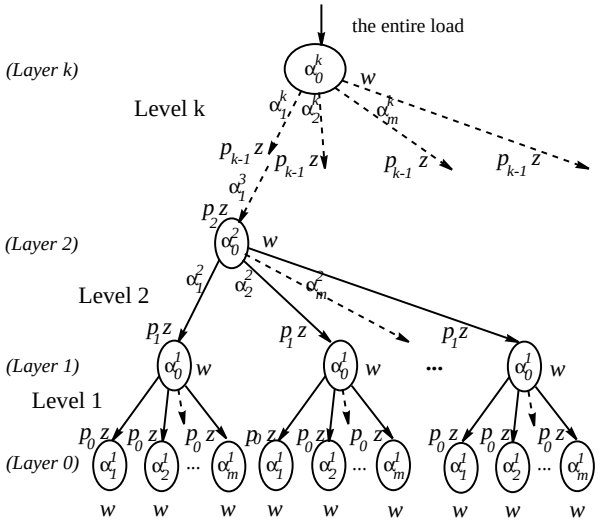

Figure 1: Homogeneous multi-level fat tree with intelligent root.

First of all, a heterogeneous single level fat tree, level $i+1$, with intelligent root is described as follows. All the children processors are connected to the root (parent) processor via communication links. Fig. 2 shows that a intelligent root processes a fraction of the load as well as distributes the remaining load to its children processors.

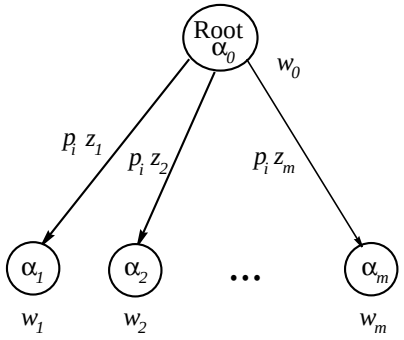

Figure 2: A heterogeneous single level fat tree, level $i+1$, with intelligent root.

Note that each child processor starts computing and transmitting immediately after receiving its assigned fraction of load and continues without any interruption until all its assigned load fraction have been processed. This is a "store and forward" mode of operation for computation and communication. The root can begin processing at time 0, the time when all the load is assumed to be present at the root.

The notations for a single heterogeneous tree are

α_0 : The load fraction assigned to the root processor.

α_i : The load fraction assigned to the i^{th} link-processor pair.

w_i : The inverse of the computing speed of the i^{th} processor.

z_i : The inverse of the link speed of the i^{th} 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 : The finish time. Time at which the last processor accomplishes computation.

Therefore, $\alpha_i w_i T_{cp}$ is the time to process the fraction α_i of the entire load on the ith processor. Note that the units of $\alpha_i w_i T_{cp}$ are $[load] \times [sec/load] \times [dimensionless\ quantity]$.

For a multi-level homogeneous fat tree, the notations are

α_0^j : The load fraction assigned to the root processor of an equivalent j^{th} level tree.

α_i^j : The load fraction assigned to the i^{th} link-processor pair on an equivalent j^{th} level tree.

w_{eq_i} : The inverse of the equivalent computing speed of the i^{th} level tree(from level i descending to level 1).

p_i : The multiplier of the inverse of expanded capacity of the links of level $i+1$ with respect to the inverse of capacity of the links on level 1. The value of the multiplier, p_i, is the inverse of the total number of children processors descended from this link. In other words, $p_i = (\sum_{j=0}^{i} m^j)^{-1}$, and $0 < p_i \leq 1$.

The following assumptions are initially made:

• The interconnection network used is a star network (single level tree network).

• The computing and communication loads are divisible (i.e. perfectly partitioned with no precedence constraints [1]).

• Transmission and computation time are proportional (linear) to the size of the problem.

• Each node transmits load concurrently, (simultaneously), to its children.

• Store and forward is the method of transmission from level to level.

III. Intelligent Root Fat Tree

• Single Level Tree, Level $i+1$, with Intelligent Root

A single level tree network, level $i+1$, with intelligent root, which has $m+1$ processors and m links, (see Fig. 2). 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 concurrently. Each processor begins computing immediately after receiving its assigned fraction of load 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 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 Fig. 3. Note that this is a completely deterministic model.

From the timing diagram, Fig. 3, an equation for the root and 1^{st} child's solution time is

$$\alpha_0 w_0 T_{cp} = \alpha_1 p_i z_1 T_{cm} + \alpha_1 w_1 T_{cp} \qquad (1)$$

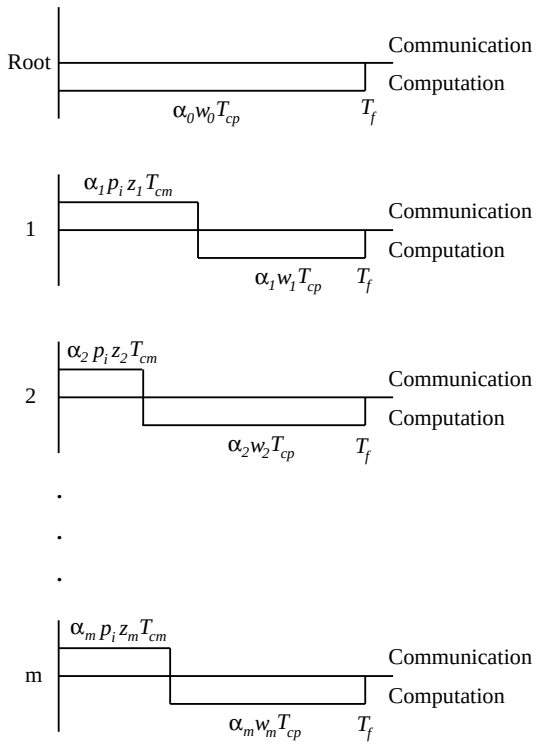

Figure 3: Timing diagram of single level fat tree, level $i+1$, with intelligent root.

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

$$\begin{aligned} & \alpha_1 p_i z_1 T_{cm} + \alpha_1 w_1 T_{cp} \\ = \quad & \alpha_2 p_i z_2 Tcm + \alpha_2 w_2 T_{cp} \end{aligned} \tag{2}$$

$$\begin{aligned} & \alpha_{i-1} p_i z_{i-1} T_{cm} + \alpha_{i-1} w_{i-1} T_{cp} \\ = \quad & \alpha_i p_i z_i T_{cm} + \alpha_i w_i T_{cp} \end{aligned} \tag{3}$$

$$\begin{aligned} & \alpha_{m-1} p_i z_{m-1} T_{cm} + \alpha_{m-1} w_{m-1} T_{cp} \\ = \quad & \alpha_m p_i z_m T_{cm} + \alpha_m w_m T_{cp} \end{aligned} \tag{4}$$

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

$$\alpha_0 + \alpha_1 + \alpha_2 + ... + \alpha_m = 1 \tag{5}$$

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

For a multi-level fat tree with intelligent root (see Fig. 1), the normalization equation for each level j (equivalent to a single level tree) is

$$\alpha_0^j + \alpha_1^j + \alpha_2^j + ... + \alpha_m^j = 1 \qquad j = 1, 2, ... \tag{6}$$

Here α_i^j is the fraction of load which one of layer j's processor (one root node in level j) distributes to the i^{th} child processor.

Now, one can manipulate equation (2) - (4) to yield the solution,

$$\alpha_i = (\frac{p_i z_{i-1} T_{cm} + w_{i-1} T_{cp}}{p_i z_i T_{cm} + w_i T_{cp}}) \alpha_{i-1} \qquad i = 2, 3, ..., m \tag{7}$$

Let

$$f_{i-1} = \frac{p_i z_{i-1} T_{cm} + w_{i-1} T_{cp}}{p_i z_i T_{cm} + w_i T_{cp}} \tag{8}$$

then

$$\alpha_i = f_{i-1} \alpha_{i-1} = \left(\prod_{j=1}^{i-1} f_j \right) \alpha_1 \tag{9}$$

$$= (\frac{p_i z_1 T_{cm} + w_1 T_{cp}}{p_i z_i T_{cm} + w_i T_{cp}}) \alpha_1 \qquad i = 2, 3, ..., m \tag{10}$$

From equation (8), $\prod_{l=1}^{k} f_l$ can be simplified as

$$\prod_{l=1}^{k} f_l = \frac{p_i z_1 T_{cm} + w_1 T_{cp}}{p_i z_{k+1} T_{cm} + w_{k+1} T_{cp}} \qquad k = 1, 2, ..., m-1 \tag{11}$$

In order to solve the set of equations, q_i is defined as:

$$q_i = \frac{w_0 T_{cp}}{p_i z_1 T_{cm} + w_1 T_{cp}} = \frac{\alpha_1}{\alpha_0} \tag{12}$$

If one substitutes this equation into the normalization equation, the normalization equation becomes

$$\frac{1}{q_i} \alpha_1 + \alpha_1 + f_1 \alpha_1 + \cdots + f_1 f_2 \ldots f_{m-1} \alpha_1 = 1 \tag{13}$$

Utilizing equation (11) and solving once again for α_1:

$$\alpha_1 = \frac{1}{\frac{1}{q_i} + 1 + \sum_{k=1}^{m-1} (\prod_{l=1}^{k} f_l)} =$$

$$\frac{1}{\frac{1}{q_i} + 1 + (p_i z_1 T_{cm} + w_1 T_{cp}) \times \sum_{k=1}^{m-1} (\frac{1}{p_i z_{k+1} T_{cm} + w_{k+1} T_{cp}})}$$

$$= \frac{1}{\frac{1}{q_i} + (p_i z_1 T_{cm} + w_1 T_{cp}) \times \sum_{k=0}^{m-1} (\frac{1}{p_i z_{k+1} T_{cm} + w_{k+1} T_{cp}})}$$

Accordingly,

$$\alpha_0 = \frac{\frac{1}{q_i}}{\frac{1}{q_i} + (p_i z_1 T_{cm} + w_1 T_{cp}) \times \sum_{k=0}^{m-1} (\frac{1}{p_i z_{k+1} T_{cm} + w_{k+1} T_{cp}})} \tag{14}$$

More generally, if we define $\prod_{j=1}^{0} f_j = 1$, then

$$\alpha_i = \frac{\prod_{j=1}^{i-1} f_j}{\frac{1}{q_i} + (p_i z_1 T_{cm} + w_1 T_{cp}) \times \sum_{k=0}^{m-1} (\frac{1}{p_i z_{k+1} T_{cm} + w_{k+1} T_{cp}})} \tag{15}$$

for $i = 1, 2, \ldots, m$.

From Fig. 4, the finish time at which a solution is achieved as:

$$T_{f,m} = \alpha_1 (p_i z_1 T_{cm} + w_1 T_{cp}) =$$

$$\frac{p_i z_1 T_{cm} + w_1 T_{cp}}{\frac{1}{q_i} + (p_i z_1 T_{cm} + w_1 T_{cp}) \times \sum_{k=0}^{m-1} (\frac{1}{p_i z_{k+1} T_{cm} + w_{k+1} T_{cp}})} \tag{16}$$

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 $i = 1, 2, \ldots, m$). Therefore, from (8), f_i is equal to 1, (for $i = 1, 2, ..., m-1$). Note for the root w_0 can be different from w_i.

For a single level tree, let $T^h_{f,0}$ be the solution time for the entire divisible load solved on the root processor and let $T^h_{f,m}$ be the solution time solved on the whole tree.

$$\begin{aligned} T^h_{f,0} &= \alpha_0 w_0 T_{cp} \qquad \text{Here, } \alpha_0 = 1 \\ T^h_{f,m} &= (\frac{1}{\frac{1}{q_i} + m})(p_i z T_{cm} + w T_{cp}) \end{aligned} \tag{17}$$

Consequently,

$$\begin{aligned} Speedup &= \frac{T^h_{f,0}}{T^h_{f,m}} = \frac{w_0 T_{cp}}{p_i z T_{cm} + w T_{cp}} (\frac{1}{q_i} + m) \\ &= q_i(\frac{1}{q_i} + m) = 1 + q_i m \end{aligned} \tag{18}$$

Here, *speedup* is the effective processing gain in using $m+1$ processors. Our finding is that the speedup of the single level homogeneous tree is equal to $\Theta(m)$, which is proportional to the number of children, per node m. Speedup is linear as long as the root CPU can concurrently (simultaneously) transmit load to all of its children. That is, the speedup of the single level tree does not saturate (in contrast to the sequential load distribution as in [1]).

•Multiple Level Fat Tree with Intelligent Root

Consider a homogeneous multi-level fat tree network where all processors have the same inverse computing speed, w, and links of level $i+1$ have the transmission speed, $p_i z$, (see Fig. 1).

$$p_i z = \left[\left(\sum_{j=0}^{i} m^j \right)^{-1} \right] z \tag{19}$$

The process of load distribution for the multi-level fat tree network using store and forward strategy for computing and communicating can be represented by Gantt chart-like timing diagrams, (see Fig. 4).

For the lowest single level tree, level 1, (see Fig. 5), the inverse computational speed of an equivalent processor is defined as w_{eq_1}. This is a valid concept as the model is a linear one (as in a Norton's equivalent queue, for instance [16]). Therefore, from equation (12) and (17), the computation time of level 1 is :

$$w_{eq_1} T_{cp} = \frac{p_0 z T_{cm} + w T_{cp}}{\frac{1}{q_0} + m} \tag{20}$$

for $q_0 = wT_{cp}/(p_0 z T_{cm} + wT_{cp})$.

Let $\sigma = zT_{cm}/wT_{cp}$, then

$$\frac{1}{q_0} = 1 + p_0 \sigma \tag{21}$$

If w_{eq_0} is defined as w, γ_0 can be defined as $w_{eq_0}/w = 1$. Hence, equation (21) can be transformed to

$$\frac{1}{q_0} = 1 + p_0 \sigma = \gamma_0 + p_0 \sigma \tag{22}$$

The goal here is to find an expression for an equivalent processor that has the same load processing characteristics as the entire homogeneous fat tree. Our strategy is to replace each of the lowest most single level tree networks (which we call level 1) with an equivalent processor. Proceeding recursively up the tree, we replace each of the current lowest most single level subtrees with an equivalent processor. This continues until the entire homogeneous fat tree network is replaced by

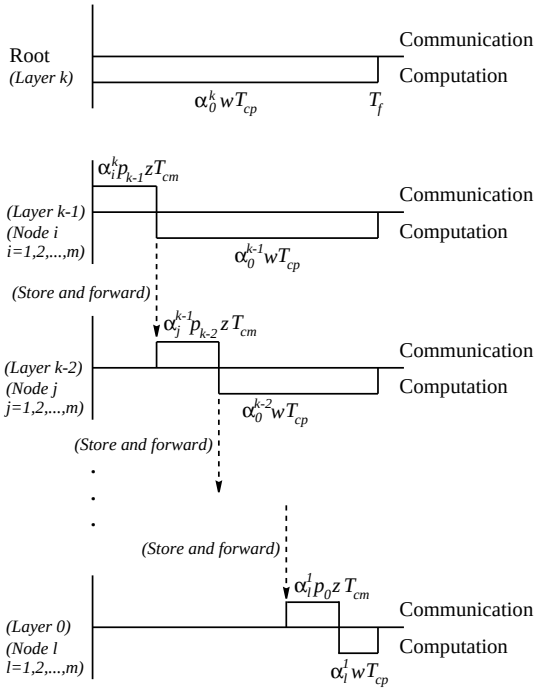

Figure 4: Timing diagram of multi-level fat tree using store and forward strategy

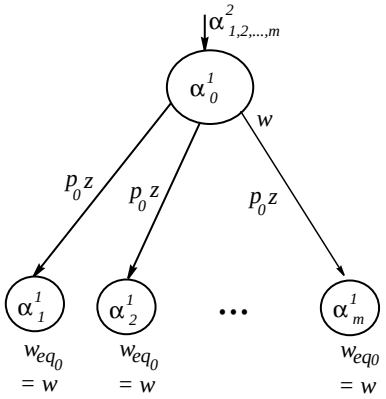

Figure 5: Level 1 of multi-level fat tree with intellginent root.

a single equivalent processor, with inverse processing speed w_{eq_k}. Here, k is the k^{th} level. Levels here are numbered from the bottom level upwards. In terms of notation, this is done from level 1 (this is the two bottom most layers), level 2 (currently next bottom most two layers), up to the top level (top two layers), (see Fig. 1).

Note that for the entire initial ($1st$) level equivalent processor replacement, both parent and children processors have the same inverse speed w, (see Fig. 5). At the k^{th} level, (equivalent to a single level tree), the parent will have inverse speed, w, and its children will have equivalent speed $w_{eq_{k-1}}$, (see Fig. 6). Referring to equation (20) and (22), the equivalent

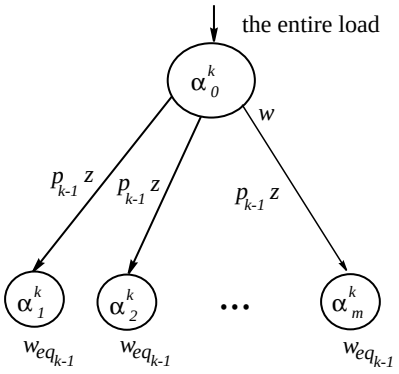

Figure 6: Level k of multi-level fat tree with intelligent root.

computation time for the $1st$ level can be defined as:

$$w_{eq_1} T_{cp} = \frac{p_0 z T_{cm} + w T_{cp}}{m + \gamma_0 + p_0 \sigma} \tag{23}$$

For level 2, (see Fig. 7), the equivalent inverse computational speed is defined as w_{eq_2}. Therefore, from equation (17), the

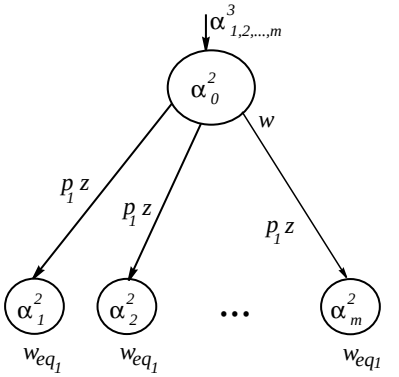

Figure 7: Level 2 of multi-level fat tree with intelligent root.

computation time

$$w_{eq_2} T_{cp} = \frac{p_1 z T_{cm} + w_{eq_1} T_{cp}}{\frac{1}{q_1} + m} \tag{24}$$

Here, from equation (12), $w_0 = w$,and $w_1 = w_2 = \cdots = w_m = w_{eq_1}$,

$$q_1 = \frac{w T_{cp}}{p_1 z T_{cm} + w_{eq_1} T_{cp}} \tag{25}$$

Let $\gamma_1 = w_{eq_1}/w$, then

$$\frac{1}{q_1} = \frac{w_{eq_1}}{w} + p_1 \sigma = \gamma_1 + p_1 \sigma \tag{26}$$

Referring to equation (24), the equivalent computation time of level 2 is given as follows:

$$w_{eq_2} T_{cp} = \frac{p_1 z T_{cm} + w_{eq_1} T_{cp}}{m + \gamma_1 + p_1 \sigma} \tag{27}$$

Therefore, the equivalent equation of a k^{th} level subtree, (see Fig. 1), the equivalent computation time

$$w_{eq_k} T_{cp} = \frac{p_{k-1} z T_{cm} + w_{eq_{k-1}} T_{cp}}{m + \gamma_{k-1} + p_{k-1} \sigma} \tag{28}$$

Referring to equation (28),

$$\begin{aligned} \gamma_k &= \frac{w_{eq_k}}{w} = \frac{w_{eq_k} T_{cp}}{w T_{cp}} \\ &= \frac{\gamma_{k-1} + p_{k-1}\sigma}{m + \gamma_{k-1} + p_{k-1}\sigma} \qquad (29) \\ &= \frac{\gamma_{k-1} + (\sum_{j=0}^{k-1} m^j)^{-1}\sigma}{m + \gamma_{k-1} + (\sum_{j=0}^{k-1} m^j)^{-1}\sigma} \qquad (30) \end{aligned}$$

Consequently, γ_k is a recursive function. The value, $1/\gamma_k$, is the speedup of a multi-level fat tree network with concurrent load distribution on each level and with store and forward computation and communication from level to level.

Let $T_{f,0}^e$ be the equivalent solution time for the entire divisible load solved on only one processor and let $T_{f,m}^{e,k}$ be the equivalent solution time of a whole homogeneous k-level fat tree network, on which each level has m children processors as well as the root processor. Then,

$$\begin{aligned} T_{f,0}^e &= 1 \cdot w T_{cp} \qquad \text{the entire load} = 1 \\ T_{f,m}^{e,k} &= 1 \cdot w_{eq_k} T_{cp} \quad \text{the entire load} = 1 \end{aligned}$$

Consequently,

$$\begin{aligned} Speedup &= \frac{T_{f,0}^e}{T_{f,m}^{e,k}} = \frac{w T_{cp}}{w_{eq_k} T_{cp}} = \frac{w}{w_{eq_k}} \\ &= \frac{1}{\gamma_k} \\ &= \frac{m + \gamma_{k-1} + (\sum_{j=0}^{k-1} m^j)^{-1}\sigma}{\gamma_{k-1} + (\sum_{j=0}^{k-1} m^j)^{-1}\sigma} \qquad (31) \\ &= 1 + \frac{m}{\gamma_{k-1} + (\sum_{j=0}^{k-1} m^j)^{-1}\sigma} \qquad (32) \end{aligned}$$

• if $m = 1$ and $p_i = 1$, this model is the same as an linear network with store and forward strategy.

• if $m = 2$, this model is a binary fat tree. If $m = 3$, this model is a ternary fat tree.

• if $p_i = 1$, this model is not a fat tree. Each link in this model has the same transmission speed.

• if $(\sum_{j=0}^{i-1} m^j)^{-1}\sigma$ approaches to zero, the model approaches an ideal case. Each node can receive the load instantly and compute the data immediately. In such assumption, the recursive function (30) can be simplified as

$$\gamma_k = \frac{\gamma_{k-1}}{m + \gamma_{k-1}} \tag{33}$$

A closed form solution

$$\gamma_k = \frac{1}{m^0 + m^1 + m^2 + \cdots + m^k} \tag{34}$$

$$Speedup = \sum_{j=0}^{k} m^i \tag{35}$$

Speedup is proportional to the total number of nodes, which is $m^0 + m^1 + m^2 + \cdots + m^k$.

Note, from (33), we can derive

$$Speedup = \frac{1}{\gamma_k} = 1 + m(\frac{1}{\gamma_{k-1}}) \tag{36}$$

This equation expresses that the speedup of k-level fat tree is the sum of the speedup of root and all the speedup from m children. The speedup of k-level equivalent tree is $\Theta(m)$, which is proportional to the number of children, per node m. The number of levels of a tree increases, the speedup will approach a linear function. Therefore, the multi-level fat tree will not saturate.

IV. Kim Type Scheduling

We note that the use of Kim type scheduling [17], where processing at a child node commences as soon as load begins to be received, can be analyzed in a similar manner to that described here. Performance should improve somewhat because of the expedited computing in this case.

V. Discussion and Conclusions

This research confirms two important points. Firstly, up to the limit of CPU speed, concurrent load distribution for a single level tree leads to a linear speedup as a function of the number of children. Secondly, the use of store and forward load distribution for a fat tree leads to a speedup which approaches a linear speedup.

Acknowledgments

The support of NSF grant CCR9912331 in the course of this research is gratefully acknowledged.

References

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

[2] Y.C. Cheng and T.G. Robertazzi, "Distributed computation with communication delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 22, pp. 60–79, 1988.

[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*, vol. 9, no. 5, pp. 429–441, May 1998.

[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*, vol. 21, no. 5, pp. 1202–1205, 1991.

[5] V. Bharadwaj, D. Ghosee, 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.

[6] V. Bharadwaj, D. Ghosee, and V. Mani, "An efficient load distribution strategy for a distributed linear network of processors with communication delays," *Computers and Mathematics with Applications*, vol. 29, no. 9, pp. 95–112, 1995.

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

[8] J. Blazewicz and M. Drozdowski, "The performance limits of a two dimensional network of load sharing processors," *Foundations of Computing and Decision Sciences*, vol. 21, no. 1, pp. 3–15, 1996.

[9] J. Blazewicz and M. Drozdowski, "Distributed processing of divisible jobs with communication start-up costs," *Discrete Applied Mathematics*, vol. 76, no. 1-3, pp. 21–41, May 1997.

[10] J. Sohn and T. G. Robertazzi, "Optimal time varying load sharing for divisible loads," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 907–924, July 1998.

[11] J. Sohn, T. G. Robertazzi, and S. Luryi, "Optimizing computing costs using divisible load analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, pp. 225–234, March 1998.

[12] S. Charcranoon, T. G. Robertazzi, and S. Luryi, "Cost efficient load sequencing in single-level tree networks," in *Proceedings 1998 Conference on Information Sciences and Systems*, Princeton, NJ, March 1998, Princeton University.

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

[14] S. Pande, D.P. Agrawal, and J. Mauney, "A scalable scheduling scheme for functional parallelism on distributedmemory multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, pp. 388–399, 1995.

[15] David E. Culler, and Jaswinder Pal Singh, "Parallel Computer Architecture: A Hardware/Software Approach," *Morgan Kaufmann Publishers*, 1999.

[16] Herzog, U., Woo, L. and Chandy, K.M., "Solution of Queueing Problems by a Recursive Technique," *IBM Journal of Research and Development*, pp. 295-300, May 1975.

[17] H.-J Kim, "A Novel Load Distribution Algorithm for Divisible Loads," *Special Issue of Cluster Computing on Divisible Load Scheduling*, Summer/Fall, 2002.