2003 Conference on Information Sciences and Systems, The Johns Hopkins University, March 12–14, 2003

Distributed Scheduling of Nonlinear Computational Loads

Jui-Tsun Hung
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, New York 11794
e-mail: trent@ece.sunysb.edu

Thomas Robertazzi[1]
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, New York 11794
e-mail: tom@ece.sunysb.edu

Abstract —

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

I. 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 [1-11] work by a number of researchers has developed algebraic means of determining the optimal fractions of total load to assign to processors and links in a given interconnection topology under a given scheduling policy. 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 [3]. 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.

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. On the other hand, 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.

In the scheduling policy used here (simultaneous start) load reception and processing may commence at the same time. Tractable load allocation and speedup equations are found for these cases, without loss of generality, for a power 2 nonlinear dependency of computation time on load size for both single level and multi-level trees topologies.

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 [2-7] computation versus communication load intensity [2-3], and numerous applications.

To evaluate a homogeneous multilevel tree, we must analyze a single level tree first. We make three major assumptions. 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.

This paper presents the types of notation and analytic background in section II. The speedup of a single level tree is obtained in section III. Furthermore, the speedup of a multi-level tree is derived in section IV. The conclusion is stated in section V.

II. Model, Notation and Theorem

• *Notation for Single Level Tree*

In this paper we assume that a node begins to process its load as soon as the load is received. This strategy is proposed by Kim [10] and we call it a 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 $w_i T_{cp}^2$ 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 from a single level tree composed of one root node and m children nodes.

$T_{f,0}$: The finish time of a processor only, (i.e. a tree without any children nodes but the root node).

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

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

Definition 2 *Speedup, the ratio of finish time on one processor (i.e. the root node) to that on an equivalent node collapsed*

[1]The support of NSF grant CCR-99-12331 is acknowledged.

from 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,

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

Finally, $(\alpha_i T_{cp})^2 w_i$ is the finish time to process the fraction α_i of the entire load on the ith processor.

- *Model and Notation for 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 Figure 1). After sub-roots receive 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".

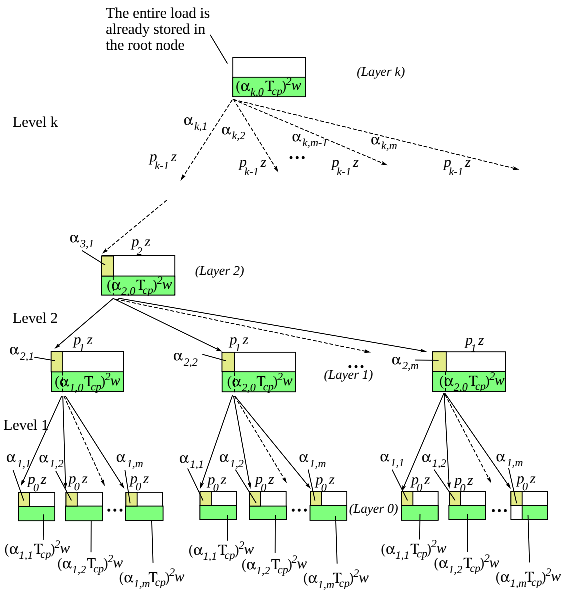

Figure 1: Structure of multi-level homogeneous tree with store and forward switching, simultaneous distribution, simultaneous start.

The notation for a multi-level 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 on the jth level subtrees.

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

$T^{h,k}_{f,m}$: 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 1).*

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 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, $p_{j-1} = p_{j-1,i}$ $(i = 1, 2, ..., m)$. Hence,

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

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

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

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,*

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

- *Nonlinear Divisible Computational Load Theorem*

If the elements in an arbitrarily divisible load are dependent on one another in some sense, the computation times based on some algorithm for each fraction of load can be nonlinear in the size of the loads (neglecting prior or post-processing). Since the elements are dependent on one other, post-processing is needed (as in sorting algorithms). 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 1 *Let an entire load be a nonempty data set S consisting of n elements. A partition of a nonempty set S is a collection of nonempty subsets that are disjoint and whose union is S (see [12], page 106). After being partitioned into $m+1$ subsets, $S_0, S_1, S_2, ..., S_m$, 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^2steps output. For a fraction, α, of load of n elements, the number of processing steps, $F(\alpha n)$, is as following,*

$$F(\alpha n) = (\alpha n)^2(steps) = \alpha^2 n^2 = \alpha^2 F(n) \qquad (1)$$

Here α is fractional number. If a fraction of load, α, is partitioned into α_1 and α_2, the number of the processing steps is invariant after the fraction load is partitioned. That is,

$$\begin{aligned} F(\alpha n) &= (\alpha n)^2 = \alpha^2 n^2 = \alpha^2 F(n) \\ &= (\alpha_1 + \alpha_2)^2 n^2 = (\alpha_1 + \alpha_2)^2 F(n) \\ &= \alpha_1^2 F(n) + \alpha_2^2 F(n) + 2\alpha_1\alpha_2 F(n) \\ &= F(\alpha_1 n) + F(\alpha_2 n) + 2\alpha_1\alpha_2 F(n) \\ &= F(\alpha_1 n) + F(\alpha_2 n) + post-processing \qquad (2) \end{aligned}$$

After the entire load is partitioned into $m+1$ subsets, and the fraction of each load is assigned as $\alpha_0, \alpha_1, \alpha_2, ..., \alpha_m$, the number of steps by a specific algorithm is

$$\begin{aligned} F(n) &= (n)^2 \\ &= [(\alpha_0 + \alpha_1 + ... + \alpha_m)n]^2 \\ &= (\alpha_0 + \alpha_1 + ... + \alpha_m)^2 \times F(n) \\ &= \left(\sum_{i=0}^{m} \alpha_i^2 + 2 \sum_{i=0,j=0}^{m} \alpha_i \cdot \alpha_j \right) \times F(n) \\ &= \sum_{i=0}^{m} F(\alpha_i n) + 2 \sum_{i=0,j=0}^{m} \alpha_i \cdot \alpha_j \times F(n) \qquad (3) \end{aligned}$$

Observing the above equation, the first term is a summation of the amount of processing steps for each fraction processed by a children 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.*

III. Processors with Simultaneous Start: Single Level Tree

When a processor uses the simultaneous start protocol, it starts processing data as soon as it receives the initial fraction of load. We assumed that the root is "intelligent" so that it can distribute load to its children while processing some fraction of the load. The simultaneous start of load reception and computation was proposed by Kim [10] and the concurrent broadcast of load over links from a root to children by Murthy and Piriyakumar [11]. Note that if load reception and computation commences simultaneously, sufficient load must be received by each point in time so a processor is not idle (starvation).

- *Single Level Tree: Root Node with Data Storage, Power χ*

In this section we find optimal load distribution formula for a power χ dependency between computation time at a node and load size at the node in a single level tree network. Simultaneous start is used. All load is available at the root at $t = 0$ (data storage case).

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.

While receiving its initial assigned fraction of load, each child processor begins computing immediately 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 [3]. The process of load distribution can be represented by Gantt chart-like timing diagrams, as illustrated in Figure 2. Note that this is a completely deterministic model.

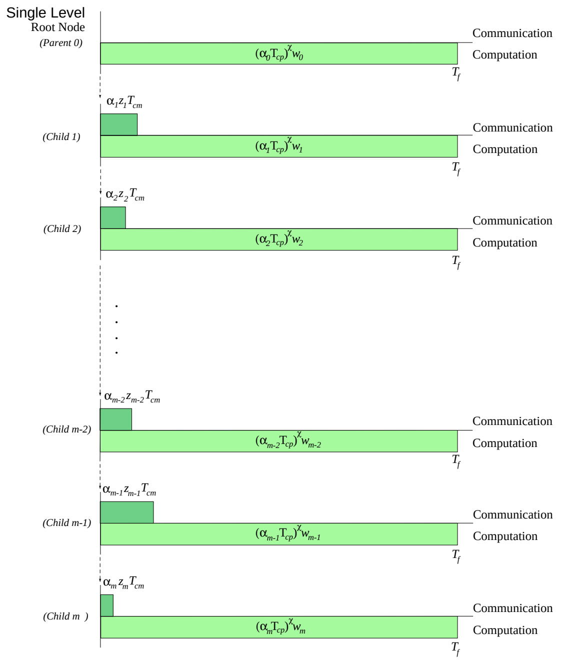
Single Level
Root Node
(Parent 0)
Communication
Computation
$(\alpha_0 T_{cp})^\chi w_0$
T_f
$\alpha_1 z_1 T_{cm}$
(Child 1)
$(\alpha_1 T_{cp})^\chi w_1$
$\alpha_2 z_2 T_{cm}$
(Child 2)
$(\alpha_2 T_{cp})^\chi w_2$
$\alpha_{m-2} z_{m-2} T_{cm}$
(Child m-2)
$(\alpha_{m-2} T_{cp})^\chi w_{m-2}$
$\alpha_{m-1} z_{m-1} T_{cm}$
(Child m-1)
$(\alpha_{m-1} T_{cp})^\chi w_{m-1}$
$\alpha_m z_m T_{cm}$
(Child m)
$(\alpha_m T_{cp})^\chi w_m$

Figure 2: Timing diagram of a single level tree with simultaneous distribution, simultaneous start, and root node with data storage.

From the timing diagram Figure 2, the fundamental recursive equations of the system can be formulated as follows:

$$(\alpha_0 T_{cp})^\chi w_0 = (\alpha_1 T_{cp})^\chi w_1 \qquad (4)$$

$$(\alpha_{i-1} T_{cp})^\chi w_{i-1} = (\alpha_i T_{cp})^\chi w_i \qquad i = 2, ..., m \qquad (5)$$

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

$$\alpha_0 + \alpha_1 + \alpha_2 + ... + \alpha_m = 1 \qquad (6)$$

This yields $m+1$ linear equations with $m+1$ unknowns. One can manipulate the recursive equations to yield a solution for the optimal allocation of load. From (4),

$$\alpha_0^\chi = \frac{w_1 T_{cp}^\chi}{w_0 T_{cp}^\chi} \alpha_1^\chi = \frac{w_1}{w_0} \alpha_1^\chi \qquad (7)$$

Let $\kappa_1 = w_0/w_1$, and then (7) becomes

$$\alpha_0^\chi = \frac{w_1}{w_0} \alpha_1^\chi = \frac{1}{\kappa_1} \alpha_1^\chi \qquad (8)$$

One obtains for integer χ [13]

$$\alpha_0 = \sqrt[\chi]{\frac{1}{\kappa_1} \alpha_1^\chi} \left(cos \frac{2h\pi}{\chi} + \sqrt{-1}\ sin \frac{2h\pi}{\chi} \right)$$

where h takes successively the values 0, 1, 2, ..., χ-1 (9)

Because $\alpha_0 \geq 0$ and α_0 is *real*, we can take $h = 0$ and obtain

$$\alpha_0 = \sqrt[\chi]{\frac{1}{\kappa_1}} \cdot \alpha_1 \qquad i = 1, 2, ..., m \qquad (10)$$

From (5),

$$\alpha_i^\chi = \frac{w_{i-1} T_{cp}^\chi}{w_i T_{cp}^\chi} \alpha_{i-1}^\chi = \frac{w_{i-1}}{w_i} \alpha_{i-1}^\chi \qquad i = 2, ..., m \qquad (11)$$

Let $\xi_i = w_{i-1}/w_i, i = 2, ..., m$. Thus, (11) becomes

$$\alpha_i^\chi = \xi_i \alpha_{i-1}^\chi \tag{12}$$

One obtains

$$\alpha_i = \sqrt[\chi]{\xi_i \alpha_{i-1}^\chi} \left(cos \frac{2h\pi}{\chi} + \sqrt{-1}\ sin \frac{2h\pi}{\chi} \right)$$

where h takes successively the values 0, 1, 2, ..., χ-1 (13)

Since $\alpha_i \geq 0$ and α_i is *real*, we can take $h = 0$ and obtain

$$\begin{aligned} \alpha_i &= \sqrt[\chi]{\xi_i} \alpha_{i-1} = \sqrt[\chi]{\prod_{l=2}^{i} \xi_l} \cdot \alpha_1 \\ &= \sqrt[\chi]{\frac{w_1}{w_i}} \cdot \alpha_1 \qquad i = 2, ..., m \end{aligned} \tag{14}$$

From the normalization equation (6) and from equation (10) and (14), we obtain the optimal fractions of load α_i as

$$\alpha_i = \frac{1}{\sqrt[\chi]{w_i} \cdot \left(\sum_{l=0}^{m} \sqrt[\chi]{\frac{1}{w_l}} \right)} \qquad i = 0, 1, 2, ..., m \tag{15}$$

From Figure 2, the finish time is expressed as follows.

$$T_{f,m} = (\alpha_0 T_{cp})^\chi w_0 \tag{16}$$

Here, $T_{f,m}$, indicates the finish time for the single divisible load solved in a single level tree, which consists of one root node as well as m children nodes. Also, $T_{f,0}$, is defined as the finish time for the entire divisible load processed on the root processor. In other words, $T_{f,0}$ is the finish time of a network composed of only one root node without any children nodes. Hence,

$$T_{f,0} = (\alpha_0 T_{cp})^\chi w_0 = (1 \times T_{cp})^\chi w_0 = T_{cp}^\chi w_0 \tag{17}$$

Now, collapsing a single level tree into a single equivalent node, one can obtain the finish time of the single level tree and the inverse of the equivalent computing speed of the equivalent node as follows.

$$T_{f,m} = (1 \times T_{cp})^\chi w_{eq} = T_{cp}^\chi w_{eq} = (\alpha_0 T_{cp})^\chi w_0 \tag{18}$$

From the Definition 1, $\gamma_{eq} = w_{eq}/w_0$, one obtains the value of γ_{eq} by (17) dividing (18). That is,

$$\gamma_{eq} = \alpha_0^\chi \tag{19}$$

Since speedup is the ratio of job solution time on one processor to job solution time on the $m+1$ processors (see Definition 2), one obtains the value of speedup from $T_{f,0}/T_{f,m}$, which is equal to $1/\gamma_{eq}$. Thus,

$$Speedup = \frac{1}{\gamma_{eq}} = \left(\frac{1}{\alpha_0} \right)^\chi = w_0 \left(\sum_{l=0}^{m} \sqrt[\chi]{\frac{1}{w_l}} \right)^\chi \tag{20}$$

Speedup is a measure of the achievable parallel processing advantage.

IV. Processors with Simultaneous Start: Homogeneous Multilevel Fat Tree Analysis, Power 2

Again, the topmost single level subtree is called level k, levels below it are generally level j, (j goes from level 1 at the bottom most level up to level $k-1$). We will derive the speedup of the whole multi-level tree by successively collapsing single level trees into equivalent nodes until the entire tree is collapsed into an equivalent node. We will first use the root without data storage model for levels, $j = 1, 2, ..., k-1$ and then use the root with data storage model for the top level, level k.

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

From Figure 3, the fundamental recursive equations of the

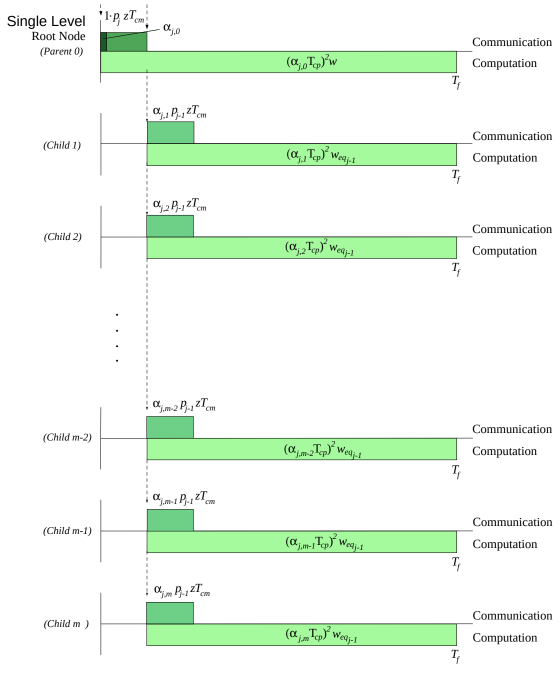
Single Level
Root Node
(Parent 0)
$1 \cdot p_j zT_{cm}$
$\alpha_{j,0}$
$(\alpha_{j,0}T_{cp})^2 w$
Communication
Computation
T_f
(Child 1)
$\alpha_{j,1} p_{j-1} zT_{cm}$
$(\alpha_{j,1}T_{cp})^2 w_{eq_{j-1}}$
(Child 2)
$\alpha_{j,2} p_{j-1} zT_{cm}$
$(\alpha_{j,2}T_{cp})^2 w_{eq_{j-1}}$
(Child m-2)
$\alpha_{j,m-2} p_{j-1} zT_{cm}$
$(\alpha_{j,m-2}T_{cp})^2 w_{eq_{j-1}}$
(Child m-1)
$\alpha_{j,m-1} p_{j-1} zT_{cm}$
$(\alpha_{j,m-1}T_{cp})^2 w_{eq_{j-1}}$
(Child m)
$\alpha_{j,m} p_{j-1} zT_{cm}$
$(\alpha_{j,m}T_{cp})^2 w_{eq_{j-1}}$

Figure 3: Timing diagram of level j subtree using store and forward switching, simultaneous distribution, simultaneous start, and root node without data storage

jth-level tree network are

$$(\alpha_{j,0} T_{cp})^2 w = (\alpha_{j,1} T_{cp})^2 w_{eq_{j-1}} + 1 \cdot p_j z T_{cm} \tag{21}$$

$$(\alpha_{j,i-1} T_{cp})^2 w_{eq_{j-1}} = (\alpha_{j,i} T_{cp})^2 w_{eq_{j-1}} \qquad i = 2, 3, ..., m \tag{22}$$

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

$$\alpha_{j,0} + \alpha_{j,1} + \alpha_{j,2} + ... + \alpha_{j,m} = 1 \tag{23}$$

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

From (21),

$$\alpha_{j,1}^2 = \frac{w T_{cp}^2}{w_{eq_{j-1}} T_{cp}^2} \alpha_{j,0}^2 - \frac{p_j z T_{cm}}{w_{eq_{j-1}} T_{cp}^2} \tag{24}$$

$$= \frac{w}{w_{eq_{j-1}}} \alpha_{j,0}^2 - \frac{p_j w}{w_{eq_{j-1}}} \cdot \frac{z T_{cm}}{w T_{cp}} \cdot \frac{1}{T_{cp}} \tag{25}$$

Here, as Definition 4, $w/w_{eq_{j-1}} = 1/\gamma_{j-1}$ and we let

$$\varsigma \quad = \quad \frac{zT_{cm}}{wT_{cp}} \cdot \frac{1}{T_{cp}} \tag{26}$$

Equation (25) becomes

$$\alpha_{j,1}^2 \quad = \quad \frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - \frac{1}{\gamma_{j-1}}p_j\varsigma \tag{27}$$

Hence,

$$\alpha_{j,1} \quad = \quad \pm\sqrt{\frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - \frac{1}{\gamma_{j-1}}p_j\varsigma} \tag{28}$$

Since $\alpha_{j,1} > 0$, we take the positive value. Therefore,

$$\alpha_{j,1} \quad = \quad \sqrt{\frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - \frac{1}{\gamma_{j-1}}p_j\varsigma} \tag{29}$$

From (22),

$$\alpha_{j,i} \quad = \quad \pm\alpha_{j,i-1} \qquad i = 2, 3, ..., m$$

Also, since $\alpha_{j,i} > 0$, $i = 2, 3, ..., m$, we take the positive value. Therefore,

$$\alpha_{j,i} \quad = \quad \alpha_{j,1} \qquad i = 2, 3, ..., m$$

The normalization equation (23) becomes

$$\begin{aligned} \alpha_{j,0} + \alpha_{j,1} + \sum_{i=2}^{m}\alpha_{j,i} &= 1 \\ \alpha_{j,0} + m\alpha_{j,1} &= 1 \end{aligned} \tag{30}$$

$$1 - \alpha_{j,0} = m\alpha_{j,1} = m\sqrt{\frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - \frac{1}{\gamma_{j-1}}p_j\varsigma} \tag{31}$$

Squaring both sides in (31), one obtains

$$(1 - \alpha_{j,0})^2 = \left(m\sqrt{\frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - \frac{1}{\gamma_{j-1}}p_j\varsigma}\right)^2 \tag{32}$$

$$1 - 2\alpha_{j,0} + \alpha_{j,0}^2 = m^2\frac{1}{\gamma_{j-1}}\alpha_{j,0}^2 - m^2\frac{1}{\gamma_{j-1}}p_j\varsigma \tag{33}$$

$$\left(\frac{m^2}{\gamma_{j-1}} - 1\right)\alpha_{j,0}^2 + 2\alpha_{j,0} - \left(1 + m^2\frac{p_j\varsigma}{\gamma_{j-1}}\right) = 0 \tag{34}$$

Finally, one obtains the value of $\alpha_{j,0}$ as follows:

$$\alpha_{j,0} \quad = \quad \frac{-1 \pm \sqrt{1 + \left(1 + m^2\frac{p_j\varsigma}{\gamma_{j-1}}\right)\left(\frac{m^2}{\gamma_{j-1}} - 1\right)}}{\frac{m^2}{\gamma_{j-1}} - 1} \tag{35}$$

Because γ_{j-1} is usually much smaller than m; thus, we found $\frac{m^2}{\gamma_{j-1}} - 1 > 0$. Now $\alpha_{j,0} > 0$ and $\frac{m^2}{\gamma_{j-1}} - 1 > 0$, and then

$$\begin{aligned} \alpha_{j,0} \quad &= \quad \frac{-1 + \sqrt{1 + \left(1 + m^2\frac{p_j\varsigma}{\gamma_{j-1}}\right)\left(\frac{m^2}{\gamma_{j-1}} - 1\right)}}{\frac{m^2}{\gamma_{j-1}} - 1} \\ &= \quad \frac{-1 + \sqrt{\frac{m^4}{\gamma_{j-1}^2}p_j\varsigma - \frac{m^2}{\gamma_{j-1}}p_j\varsigma + \frac{m^2}{\gamma_{j-1}}}}{\frac{m^2}{\gamma_{j-1}} - 1} \end{aligned} \tag{36}$$

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

$$\begin{aligned} &T_{f,m}^{h,j} = (1 \cdot T_{cp})^2 w_{eq_j} = (\alpha_{j,0}T_{cp})^2 w \\ &\text{where } j = 1, 2, ..., k-1 \end{aligned} \tag{37}$$

From Definition 4,

$$\begin{aligned} \gamma_j \quad &= \quad \frac{w_{eq_j}}{w} = \alpha_{j,0}^2 \\ &= \quad \frac{1}{\left(\frac{m^2}{\gamma_{j-1}} - 1\right)^2} \times \Bigg\{1 + \frac{m^4}{\gamma_{j-1}^2}p_j\varsigma - \frac{m^2}{\gamma_{j-1}}p_j\varsigma + \frac{m^2}{\gamma_{j-1}} \\ &\quad -2\sqrt{\frac{m^4}{\gamma_{j-1}^2}p_j\varsigma - \frac{m^2}{\gamma_{j-1}}p_j\varsigma + \frac{m^2}{\gamma_{j-1}}}\ \Bigg\} \\ &\quad \text{where } j = 1, 2, ..., k-1 \end{aligned} \tag{38}$$

- *Level k Subtree: Root Node with Data Storage, Power 2*

The timing diagram of the top equivalent single level tree, level k, is similar to Figure 2. However, the following notation is replaced as $\alpha_i = \alpha_{k,i}$ $(i = 0, 1, 2, ..., m)$; $z_i = p_{k-1}z$ $(i = 1, 2, ..., m)$; $w_0 = w$; and $w_i = w_{eq_{k-1}}$ $(i = 1, 2, ..., m)$. Consequently, the fundamental recursive equations of the kth-level subtree are derived as follows.

$$(\alpha_{k,0}T_{cp})^2 w = (\alpha_{k,i}T_{cp})^2 w_{eq_{k-1}} \qquad i = 1, 2, 3, ..., m \tag{39}$$

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

$$\alpha_{k,0} + \alpha_{k,1} + \alpha_{k,2} + ... + \alpha_{k,m} = 1 \tag{40}$$

This gives $m+1$ linear equations with $m+1$ unknowns. From (39),

$$\alpha_{k,i}^2 = \frac{wT_{cp}^2}{w_{eq_{k-1}}T_{cp}^2}\alpha_{k,0}^2 = \frac{1}{\gamma_{k-1}}\alpha_{k,0}^2 \tag{41}$$

$$\alpha_{k,i} = \pm\sqrt{\frac{1}{\gamma_{k-1}}}\alpha_{k,0} \tag{42}$$

Since $\alpha_{k,i} > 0$,

$$\alpha_{k,i} = \frac{1}{\sqrt{\gamma_{k-1}}}\alpha_{k,0} \qquad i = 1, 2, ..., m \tag{43}$$

From (40) and (43), one can derive the distribution $\alpha_{k,0}$ as follows:

$$\begin{aligned} \alpha_{k,0} + \sum_{i=1}^{m}\frac{1}{\sqrt{\gamma_{k-1}}}\alpha_{k,0} \quad &= \quad 1 \\ \alpha_{k,0} = \frac{1}{\frac{m}{\sqrt{\gamma_{k-1}}} + 1} \end{aligned} \tag{44}$$

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

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

From Definition 4,

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

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

For a *homogeneous multi-level fat tree using simultaneous start*, the computation capability of each node is w, thus, w_{eq_0} is equal to w, which is the inverse computing speed of the bottom most layer nodes. To summarize,

$$\gamma_0 = \frac{w_{eq_0}}{w} = \frac{w}{w} = 1 \quad (45)$$

$$\gamma_j = \frac{1}{\left(\frac{m^2}{\gamma_{j-1}} - 1\right)^2} \times \left\{ 1 + \frac{m^4}{\gamma_{j-1}^2} p_j\varsigma - \frac{m^2}{\gamma_{j-1}} p_j\varsigma + \frac{m^2}{\gamma_{j-1}} - 2\sqrt{\frac{m^4}{\gamma_{j-1}^2} p_j\varsigma - \frac{m^2}{\gamma_{j-1}} p_j\varsigma + \frac{m^2}{\gamma_{j-1}}} \right\}$$

$$\text{where } j = 1, 2, ..., k-1 \quad (46)$$

$$\gamma_k = \frac{1}{\left(\frac{m}{\sqrt{\gamma_{k-1}}} + 1\right)^2} \quad (47)$$

$$Speedup = \left(\frac{m}{\sqrt{\gamma_{k-1}}} + 1\right)^2 \quad (48)$$

- *Special Case: $p_j\varsigma$ Approaches Zero*

If $p_j\varsigma \rightarrow 0$, the model will approach an ideal case. Each node can receive the load instantly and compute the data immediately.

Since the structure of this multilevel tree can be collapsed from bottom most level subtree upwards to topmost level subtree and finally collapsed into a single node, the recursive equation (46) can be simplified as

$$\gamma_j = \frac{1 + \frac{m^2}{\gamma_{j-1}} - 2\sqrt{\frac{m^2}{\gamma_{j-1}}}}{\left(\frac{m^2}{\gamma_{j-1}} - 1\right)^2} = \frac{1}{\left(\frac{m}{\sqrt{\gamma_{j-1}}} + 1\right)^2} \quad (49)$$

Since the formula of γ_j is the same form as that of γ_k, we can obtain a closed form solution of γ_k.

$$\gamma_k = \frac{1}{(m^0 + m^1 + m^2 + \cdots + m^k)^2} \quad (50)$$

$$Speedup = \sum_{l=0}^{k} (m^l)^2 \qquad \text{where } k = 1, 2, 3, ... \quad (51)$$

Speedup, which is equal to $(m^0 + m^1 + m^2 + \cdots + m^k)^2$, is proportional to the total number of nodes by power 2. This is consistent with the ideal situation.

V. Conclusion

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.

This work indicates that an (integer) χth order dependency of computation time on divisible problem size necessitates the solution of an χth order polynomial. It should be noted that numerical (arithmetic) problem may occur in solving polynomials when χ is large.

References

[1] V. Bharadwaj, D. Ghose, and T. Robertazzi, "A new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, pp. 7–18, Jan 2003.

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

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

[4] J. T. Hung, H. Y. Kim, and T. G. Robertazzi, "Scalable scheduling in parallel processors," *2002 Conference on Information Sciences and Systems*, pp. 376–381, 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, 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, L. A., 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," 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, pp. 245–251, March 1998.

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

[13] R. S. Burington, *Handbook of Mathematical Tables and Formulas.* McGraw-Hill Book Company, 5 ed., 1973.