STONY BROOK UNIVERSITY

CEAS Technical Report 823

Scheduling Nonlinear Computational Loads:

Analysis and Proof

Jui Tsun Hung and Thomas G. Robertazzi

September 5, 2006

Scheduling Nonlinear Computational Loads

Jui Tsun Hung and Thomas G. Robertazzi

Abstract

A scheduling model is studied where the computing time function of each node is a nonlinear function of the size of its assigned load. Speedup and closed form solutions for optimal load allocation are obtained for simultaneous load distribution. An iterative solution technique is presented for sequential load distribution. Super-linear speedup is possible when the computing time function is a nonlinear function of the amount of fractional load assigned.

Index Terms

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

1 Introduction

It is well known that many algorithms have a computational complexity that is a nonlinear function of problem size. Such nonlinear loads and algorithms are considered here in the context of parallel processing. In this paper we use divisible load scheduling techniques in order to make analytical progress and because divisible loads are of interest in their own right. A divisible load is an input load that can be arbitrarily partitioned among processors and links in order to gain the benefits of parallel processing. No precedence relationships are assumed for the input data.

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

Jui Tsun Hung, Department of Computer Science, Stony Brook University, Stony Brook, NY 11794. Phone (631) 632-8448, E-mail: jason@cs.sunysb.edu.

We note, and discuss below, that algorithms of nonlinear complexity that assume the divisibility of the input data are different from algorithms of linear complexity in terms of a need for post-processing. That is, generally the results of nonlinear sub-problems solved on individual processors need to be integrated (post-processed) to create the overall solution. As a simple example, a large list that is to be sorted can be partitioned among N processors. The sorted sub-lists then need to be merged (post-processed) to create the final sorted list. We make the empirical observation that to some extent the need to do post-processing arises as a result of the dependent nature of the data in nonlinear problems. Unlike the situation for divisible loads of linear computational complexity, where the data elements are relatively independent of each other, the elements of the solution for nonlinear problems depends on the relationship between input data elements.

Tree networks are considered in this paper. A single level tree (star) is a fundamental interconnection topology. Multilevel tree networks can be used as a spanning distribution tree embedded in other interconnection topologies as well as being an interconnection topology of interest in itself. It is assumed that the optimal sequence of load distribution is applied in a single level tree or in subtrees of a multilevel tree [1] in order to achieve the minimum processing time. That is, the sequence of load distribution by the root or parent node should follow the order in which the children link speeds decrease.

Three representative situations involving tree networks are considered here. First, speedup and closed form solutions for optimal load allocation are found for a single level tree using simultaneous load distribution (i.e. the root can transmit load to its children concurrently). For simplicity, the computing function has a quadratic computational complexity in the amount of assigned load. Second, the optimal speedup of a multilevel layer homogeneous tree is found under simultaneous load distribution, store and forward switching, and quadratic computational complexity of the load. Finally, an iterative solution is developed for a single level tree using sequential load distribution.

It should be noted that sequential and simultaneous load distribution provide a wide variety of modeling possibilities. Sequential load distribution has been well studied for loads of linear complexity and is realistic when a root can communicate with only one child at a time. The improved performance and scalability of simultaneous distribution [2], [3] over sequential distribution motivate future server architectures where one server can distribute load on multiple outgoing links simultaneously. This fits in well with the needs of grids such as the ATLAS physics

experiments at CERN (Center European for Nuclear Research) where expensive international links need to be kept at high utilizations.

Most of the work on divisible load theory has used linear models. An exception is work by Drozdowski and Wolniewicz [4] who demonstrated super-linear speedup when processing time is a piecewise linear (and thus nonlinear) function of assigned load in the modeling of the memory hierarchy of a computer. Drozdwoski and Wolniewicz's results were obtained through the use of mathematical programming. In this paper analytic results are presented.

Divisible Load Theory Review

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], [5], [6], [2], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [3], [4], [22], [23] 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. Here optimality is defined in terms of speedup and execution times. The theory to date largely involves loads of linear computational complexity. That is, computation and communication times are proportional to the size of the load fraction assigned to a processor or link, respectively.

Divisible load modeling should be of interest as it models, both computation and network communication in a completely integrated manner. Moreover, it is tractable with its linearity assumption. Optimal divisible load scheduling has been developed for various interconnection topologies [14], such as linear daisy chains [6], buses [8], trees [7], [15], hypercubes [9], and two and three dimensional meshes [16], [17]. A number of scheduling policies have been investigated including multi-installments [18] and multi-round scheduling [11], simultaneous distribution [2], [13], simultaneous start [12], detailed parameterizations and solution time optimization [21], and combinatorial schedule optimization [19]. Divisible loads may be divisible in fact or as an approximation as in the case of a large number of relatively small independent tasks [10]. Introductions to divisible load scheduling theory appear in [1], [5], [20].

The next section describes models and notations. Following that, the properties of computational time function are

described in section 3. The scheduling performance using store and forward switching, simultaneous distribution, and staggered start protocols is derived in section 4, for a heterogeneous single level tree, and in section 5, for a layer homogeneous multilevel tree. In these two sections the computing time function is considered a function of power 2 of the size of assigned fractional load. In section 6 the performance of a single level tree using sequential distribution is explored where the computing time function is of power χ of the size of the assigned fractional load. Finally, the conclusion and lessons learned are stated in section 7.

2 Models and Notation

In this paper only staggered start scheduling is considered. Staggered start means that load can not be processed at a node until the node has completely received its fractional load. On the other hand, if a node begins to process its fractional load as soon as the load is received, the protocol is called simultaneous start [12] (this is not discussed for reasons of space). In the following sections a single level tree using simultaneous distribution (section 4) and using sequential distribution (section 6) is modeled. Simultaneous distribution, where load is transmitted concurrently over multiple links, was first proposed by Piriyakumar and Murthy [13]. In sequential distribution a parent node can transmit fractional load to only one of its children at a time.

A heterogeneous tree is a tree with distinct computing speeds at different nodes. A layer homogeneous tree has equal computing speeds of nodes and equal communication speeds of links at the same layer in the tree.

2.1 Model and Notations for a Single Level Tree

A heterogeneous single level tree using staggered start is illustrated in Fig. 1 where each node contains a miniature timing diagram. A heterogeneous single level tree rooted at $node_{<0>}$ can be collapsed into an equivalent node, $node^{eq}_{<0>}$, whose equivalent inverse computing speed is denoted ω_0^{eq}. That is, a single level tree can be collapsed into an equivalent processor with an equivalent computing speed equal to the speed of the original single level tree. This is an important concept for a multilevel tree where one recursively collapses single level subtrees into equivalent nodes so as to obtain an equivalent processor for the entire multilevel tree network and obtain the speedup formulae for the tree. The concept of processor equivalence was introduced by Robertazzi in 1993 [1] [24].

The notation and symbols are as follows:

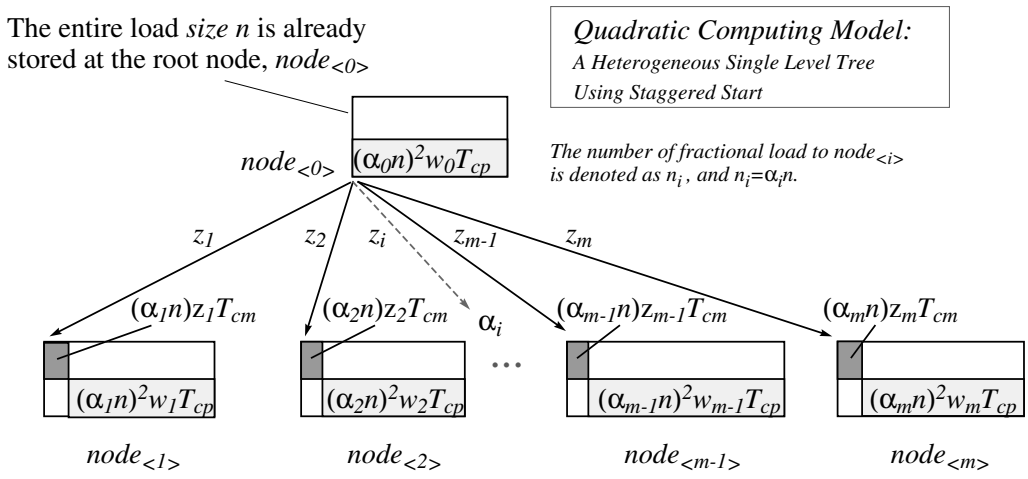

Fig. 1. A single level tree using the staggered start model. The worst-case running cost of an algorithm at $node_{<i>}$ is assumed to be $\Theta(n_i^2)$

n : The number of records (or called objects or atomic pieces) of the entire load in a tree network. It is also called the size of the entire load.

$n_i = \alpha_i n$: The number of records of a fractional load at $node_{<i>}$ (where $i = 0, 1, 2, \ldots, m$).

α_0 : The load fraction assigned to the root processor.

α_i : The load fraction assigned to the ith link-processor pair (where $i = 0, 1, 2, \ldots, m$).

w_i : The inverse computing speed at the ith processor (where $i = 0, 1, 2, \ldots, m$).

w_0^{eq} : The equivalent inverse computing speed of the equivalent node, $node_{<0>}^{eq}$, collapsed from a single level tree rooted at $node_{<0>}$.

z_i : The inverse communication speed on the ith link (where $i = 0, 1, 2, \ldots, m$).

T_{cp} : Computing intensity constant.

T_{cm} : Communication intensity constant.

T_f : The finish time. Time at which each processor accomplishes computation.

Definition 1: γ^{eq}, the ratio of the inverse computing speed at an equivalent node, $node_{<0>}^{eq}$, to that at the root node, $node_{<0>}$. The equivalent node is the result of collapsing a single level tree rooted at $node_{<0>}$.

$$\gamma^{eq} = w_0^{eq}/w_0 \tag{1}$$

Definition 2: $Speedup$, the ratio of the computing speed at the equivalent node to that at the root node in the tree, that is, the inverse of γ^{eq}.

$$Speedup = 1/\gamma^{eq} = \omega_0/\omega_0^{eq} \tag{2}$$

2.2 Model and Notations for a Multilevel Tree

A heterogeneous multilevel tree is not employed because of the complexity of the index system; therefore, in this paper a simpler physical model of layer homogeneous structure is evaluated (see Fig. 2). In this physical model

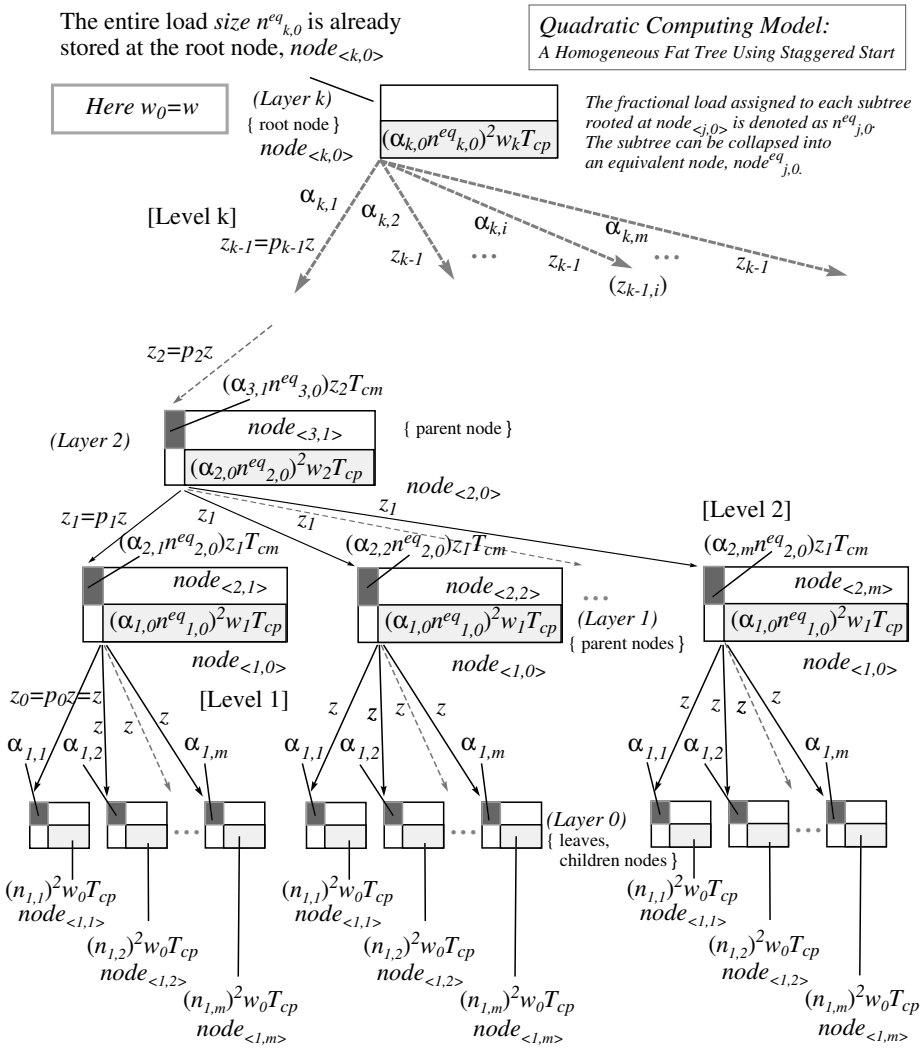

Fig. 2. A multilevel homogeneous tree using store and forward switching, simultaneous distribution and staggered start. The worst-case running cost of an algorithm at $node_{<j,0>}$ is assumed to be $\Theta(n_{j,0}^2)$ where $n_{j,0}$ is the fractional load for $node_{<j,0>}$ to process.

the root $node_{<k,0>}$ is the only node with no parent (the topmost layer) and a node with no children is called a leaf (the bottommost layer). A subtree, on the other hand, rooted at $node_{<j,0>}$ is a tree induced by its descendants of $node_{<j,0>}$ (where $0 < j < k$, and j is an integer). For off-loading communications the root and parent processors are equipped with front-end processors. The degree of $node_{<j,0>}$ is defined as m; in other words, $node_{<j,0>}$ has m children .

After a parent receives all fractional loads for the subtree rooted at itself, immediately it starts distributing fractional loads to its descendants simultaneously or sequentially, according to the policy used. The use of cut through switching for linear models is considered in [3], [23].

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

$n_{j,0}^{eq}$: The number of records of the fractional load delivered to the equivalent node, $node_{<j,0>}^{eq}$, collapsed from a subtree rooted at node, $node_{<j,0>}$ (where $j = 1, 2, \ldots, k$).

$n_{j,i} = \alpha_{j,i} n_{j,0}^{eq}$: The number of records of a fractional load processed at $node_{<j,i>}$ (where $i = 0, 1, 2, \ldots, m$, $j = 1, 2, \ldots, k$).

$\alpha_{j,0}$: The load fraction assigned to the root processor in the jth level subtree (where $j = 1, 2, \ldots, k$).

$\alpha_{j,i}$: The load fraction assigned to the ith link-processor pair in the jth level subtree (where $i = 0, 1, 2, \ldots, m$, $j = 1, 2, \ldots, k$).

w : The inverse computing speed at each leaf processor at the bottommost layer.

w_j : The inverse computing speed at each parent processor in the jth layer (where $j = 1, 2, \ldots, k$).

$w_{j-1,i}^{eq}$: The inverse computing speed at an equivalent ith node representing the $(j-1)$th level subtree, consisting of level $j-1$ descending to level 1. The equivalent node collapsed from a subtree rooted at $node_{<j-1,i>}$ is denoted $node_{<j-1,i>}^{eq}$. For simplicity, we assume that $\omega_{j-1}^{eq} = \omega_{j-1,i}^{eq}$ (where $i = 1, 2, \ldots, m$) in a layer homogeneous multilevel tree.

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

The value of the multiplier $p_{j-1,i}$ is defined here as the inverse of the total number of children processor descendants at and below level j for the ith subtree. The variable $p_{j-1,i}$ allows fat tree modeling. A fat tree

allocates more communication capacity to nodes near the root to improve the speed of load distribution and to prevent bottlenecks. In a homogeneous multilevel fat tree, we assume $p_{j-1} = p_{j-1,i}$ $(i = 1, 2, \ldots, m)$. Hence,

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

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

$z_j = p_j z$: The inverse communication speed at each link in the $j+1$st level subtree.

Definition 4: γ_j^{eq}, the ratio of the inverse computing speed at the equivalent node, $node_{<j,0>}^{eq}$, at level j to that of the root node, $node_{<j,0>}$.

$$\gamma_j^{eq} = w_j^{eq}/w_j \tag{4}$$

Definition 5: $Speedup$, the inverse of γ_j^{eq}.

$$Speedup = 1/\gamma_j^{eq} = w_j/w_j^{eq} \tag{5}$$

3 The Properties of Computational Time Function

In this section a computational time function is defined in terms of the running cost of an algorithm, the inverse of CPU speed, and the computing intensity constant. A distinction is made between data partitioned by hardware (on multiple processors) and partitioned by software (by a process on a single machine), each of which has somewhat different characteristics.

The computational time function at a node (or a single machine) is defined here as the time that it takes to process its own fractional load. On the other hand, the run time of an algorithm is sometimes defined as the number of steps [25] in the literature. This is an appropriate description for running time because the performance of an algorithm should be based on a standard which is independent of the computing powers of a variety of machines.

The optimal performance of a tree network is machine-dependent and we assume that all the fractional loads are processed (to some extent) concurrently. Therefore we define the computation time function as a product of the running cost of an algorithm (unit $step$), the inverse of node computing speed (unit $seconds/per\ step$) and the computation intensity defined earlier (unit dimensionless). In this paper, we use the running cost instead of the running time for algorithms so as to avoid confusion. The computational equation at $node_{<i>}$ (a node indexed to

i) can be expressed as

$$F_i^{cp}(\cdot) = F_i^{algm}(\cdot) \times F_i^{inv \cdot CPU \cdot sp}(\cdot) T_{cp} \tag{6}$$

The notation is described as follows.

1) $F_i^{cp}(\cdot)$ is the computational time at $node_{<i>}$ (unit $second$).
2) $F_i^{algm}(\cdot)$ is the running cost of an algorithm processing a fractional load (unit $step$).
3) $F_i^{inv \cdot CPU \cdot sp}(\cdot)$ is the inverse of CPU speed at $node_{<i>}$ (unit $seconds/per\ step$) - in other words, it is the CPU execution time for each step (or for each instruction). Therefore, it can be denoted with a conventional notation as w_i .

As a result, the computational equation (6) becomes

$$F_i^{cp}(\cdot) = F_i^{algm}(\cdot) w_i T_{cp} \tag{7}$$

For simplicity, $F^{algm}(\cdot)$ can be reduced to a function of the number of records (where a "record" is an indivisible piece of data). The size of a load is represented by the number of records (or atomic pieces).

On the other hand, we denote the communication time function $F_i^{cm}(\cdot)$ as

$$F_i^{cm}(\cdot) = Load_i(\cdot) z_i T_{cp} \tag{8}$$

Here we assume that the communication time is linearly proportional to the load size, $Load_i$.

3.1 Hardware Partition

The core method for parallel computing is partitioning a load into fractions and delivering these fractional loads to all nodes so that the assigned data can be processed concurrently. This decreases the finish time of a processed load, or improves the speedup of data processing. We call this partition a hardware partition for multiple machines, which is in contrast to a software partition on a single machine.

An "equivalent" node, an established concept [6], [24], has identical operating characteristics to the subnetwork it replaces. The final computational time of a subtree rooted at $node_{<0>}$ (or an equivalent node, $node_{<0>}^{eq}$), is equal to the sum of the computational time at the root node and the time for the root to collect and post-process the results from its children. The hardware partition cost is denoted $D_0^{hd}(\cdot)$. The cost of post-processing including collecting

and processing data at $node_{<0>}$ is denoted $C_0^{hd}(\cdot)$. Accordingly, at the root node the equivalent computational function can be expressed as

$$F_0^{cp \cdot eq}(\cdot) = F_0^{cp}(\cdot) + D_0^{hd}(\cdot) + C_0^{hd}(\cdot) \tag{9}$$

where $F_0^{cp \cdot eq}(\cdot)$ is the equivalent computation function of a subtree rooted at $node_{<0>}$ (or the computation function at the equivalent node, $node_{<0>}^{eq}$). However, at child $node_{<i>}$ the equivalent computational function can be expressed as

$$F_0^{cp \cdot eq}(\cdot) = F_i^{cm}(\cdot) + F_i^{cp}(\cdot) \tag{10}$$

Here $F_i^{cm}(\cdot)$ is the communication time function for the root to transmit an assigned fractional load to it child $node_{<i>}$. Hence, from Equations (7), (8), (9), and (10) one obtains

$$F_0^{algm.eq}(\cdot) w_0^{eq} T_{cp} = [F_0^{algm}(\cdot) + D_0^{hd.algm}(\cdot) + C_0^{hd.algm}(\cdot)] w_0 T_{cp} \tag{11}$$

$$F_0^{algm.eq}(\cdot) w_0^{eq} T_{cp} = Load_i(\cdot) z_i T_{cp} + F_i^{algm}(\cdot) w_i T_{cp} \tag{12}$$

Here the $algm$ superscript indicates dependence on a particular algorithm. The second equation is for simultaneous load distribution, with staggered start.

Provided that the number of records of an entire load is n (where n is sufficiently large) and provided that a subtree rooted at $node_0$ has m children nodes, which is indicated as $node_{<i>}$ (where $i = 1, 2, 3, ..., m$), the hardware divide-and-conquer property can be described as follows.

Divide: The number of divide steps is constant and related to $m+1$ because there are $m+1$ nodes in a subtree as assumed. Then $D_0^{hd.algm}(n)$ is in the $\Theta(1)$ set.

Conquer: There are $m+1$ subproblems in a processing task and each node is assigned a subproblem. Besides, the size of fraction load at $node_{<i>}$ is represented as n_i ($i = 0, 1, 2, ..., m$).

Combine: The combine procedure depends on the particular algorithm used. For instance, the combine procedure of a sorting problem depends on the extent to which the records are already somewhat sorted. We assume that the output result from each node is already sorted. According to this situation, $C_0^{hd.algm}(n) = \Theta(n)$

Now equation (11) and (12) are transformed into

$$F_0^{algm.eq}(n)w_0^{eq}T_{cp} = [F_0^{algm}(n_0) + \Theta(1) + \Theta(n)]w_0T_{cp} \tag{13}$$

$$F_0^{algm.eq}(n)w_0^{eq}T_{cp} = Load_i(n_i)z_iT_{cp} + F_i^{algm}(n_i)w_iT_{cp} \tag{14}$$

Provided that the algorithms used among all nodes including the equivalent node are the same, we may let $F_0^{algm.eq}(\cdot) = F_0^{algm}(\cdot) = F^{algm}(\cdot)$ and let $Load_i(n_i) = n_i = \alpha_i n$. Finally equation (13) and (14) become

$$F^{algm}(n)w_0^{eq}T_{cp} = [F^{algm}(n_0) + \Theta(1) + \Theta(n)]w_0T_{cp} \tag{15}$$

$$F^{algm}(n)w_0^{eq}T_{cp} = n_iz_iT_{cp} + F^{algm}(n_i)w_iT_{cp} \tag{16}$$

3.2 Software Partition

The running cost of an algorithm corresponding to a software divide-and conquer approach at $node_{<i>}$, which receives n_i records, can be expressed as

$$T(n_i) = aT(n_i/b) + D(n_i) + C(n_i) \tag{17}$$

Let

$$F^{algm}(n_i) = T(n_i) \tag{18}$$

Then one obtains

$$F^{algm}(n_i) = T(n_i) = aT(n_i/b) + D(n_i) + C(n_i) \tag{19}$$

The software divide-and conquer properties are illustrated as follow.

Divide: The process of divide steps takes only constant time. Because the data processing problem is divided into b computational subproblems, this leads $D(n_i)$ in the set of $\Theta(1)$.

Conquer: Generally a subproblems with the size n_i/b are solved recursively.

Combine: If the combine procedure at $node_{<i>}$ has n_i records, the combining cost is denoted as $C(n_i)$. For instance, if the algorithm is a sorting process, it takes $\Theta(n_i)$ steps.

Note here we use a rather than b to be more general. As an example, the sorting problem has the running cost formula as follows.

$$F^{algm}(n_i) = T(n_i) = aT(n_i/b) + \Theta(1) + \Theta(n_i) \tag{20}$$

$T(n)$ can be of the order of growth $n \log n$, n^2, n^3, 2^n, or $n!$, and so on.

3.3 Applications

Two categories of applications are illustrated as follows.

1) *Linear Applications:* Provided that the running cost is linear to the number of records, $T(n)$ is in the set $\Theta(n)$. Because of the linearity property, the output of each record is independent of those of other records after data processing. Therefore the post-processing cost $C_0^{hd.algm}(\cdot)$ is equal to zero. For simplicity, let $F^{algm}(n_i) = \Theta(n_i) = n_i$. Because the size of the fraction of the load n_i is equal to $\alpha_i n$, equation (15) and (16) become

$$n \times w_0^{eq} T_{cp} = [\alpha_0 n + \Theta(1)] w_0 T_{cp} \tag{21}$$

$$n \times w_0^{eq} T_{cp} = \alpha_i n z_i T_{cp} + \alpha_i n w_i T_{cp} \tag{22}$$

If the number of records is sufficiently large, equation (22) reduces to

$$w_0^{eq} T_{cp} = \alpha_0 w_0 T_{cp} \tag{23}$$

$$w_0^{eq} T_{cp} = \alpha_i z_i T_{cp} + \alpha_i w_i T_{cp} \tag{24}$$

2) *Nonlinear Applications:* As an example, provided that $T(n) = \Theta(n^2)$, then $F^{algm}(n_i) = \Theta(n_i^2)$.

For simplicity, let $F^{algm}(n_i)$ be n_i^2, equation (15) and (16) become

$$(n)^2 w_0^{eq} T_{cp} = [(\alpha_0 n)^2 + \Theta(1) + \Theta(n)] w_0 T_{cp} \tag{25}$$

$$(n)^2 w_0^{eq} T_{cp} = (\alpha_i n) z_i T_{cp} + (\alpha_i n)^2 w_i T_{cp} \tag{26}$$

If the number of records is sufficiently large such that $(\alpha_0 n)^2 \gg \Theta(n)$, then equation (25) is reduced to

$$(n)^2 w_0^{eq} T_{cp} \simeq (\alpha_0 n)^2 w_0 T_{cp} \tag{27}$$

Consequently, after hardware partition the equivalent computational time at $node_{<0>}$ becomes a quadratic equation in the load size, $\alpha_0 n$. Thus conditions (26) and (27), will be employed in the following two sections.

4 The Speedup Performance of a Single Level Tree Using Simultaneous Distribution

In this section we consider a heterogeneous single level tree in which processors use simultaneous load distribution and the staggered start protocol to process their assigned fractional loads. Using the staggered start protocol a processor must receive its load completely before it begins to process the load. The root node can distribute load to its children while processing some fraction of the load. In this sense the root may be considered to have a front end sub-processor for communications off-loading.

4.1 Speedup Derivation for A Single Level Tree with Running Time $\Theta(n_i^2)$

The structure of a single level tree network with $m+1$ processors and m links is illustrated in Fig. 1. All children processors are connected to the root processor via direct communication links. The root processor, assumed to be the only processor at which the divisible load arrives, partitions a total processing load optimally into $m+1$ fractions, keeps its own fraction α_0, and distributes the other fractions, $\alpha_1, \alpha_2, \ldots, \alpha_m$, to the children processors respectively and concurrently. Given that the entire load contains $n\ records$ (or $n\ atomic\ pieces$), at the root $node_{<0>}$ the fractional load is denoted n_0 (where $n_0 = \alpha_0 n$) and at child $node_{<i>}$ the fractional load is n_i (where $n_i = \alpha_i n$, $i = 1, 2, \ldots, m$).

As an example in this section we assume that the worst-case running cost of an algorithm is $\Theta(n_i^2)$ ($i = 0, 1, 2, \ldots, m$) and the computation time function at a node becomes a quadratic equation in the load size, n_i. However, the communication time function on a link is still assumed linear in load size via the link.

In order to minimize the processing finish time, all of the utilized processors in the network must finish computing at the same time [1]. Intuitively, otherwise load could be transferred from busy processors to idle processors to improve the solution (see the Appendix for a proof). The process of load distribution can be represented by Gantt chart-like timing diagrams as illustrated in Fig. 3. It is assumed that at the root node the entire load is available for distribution at time $t = 0$.

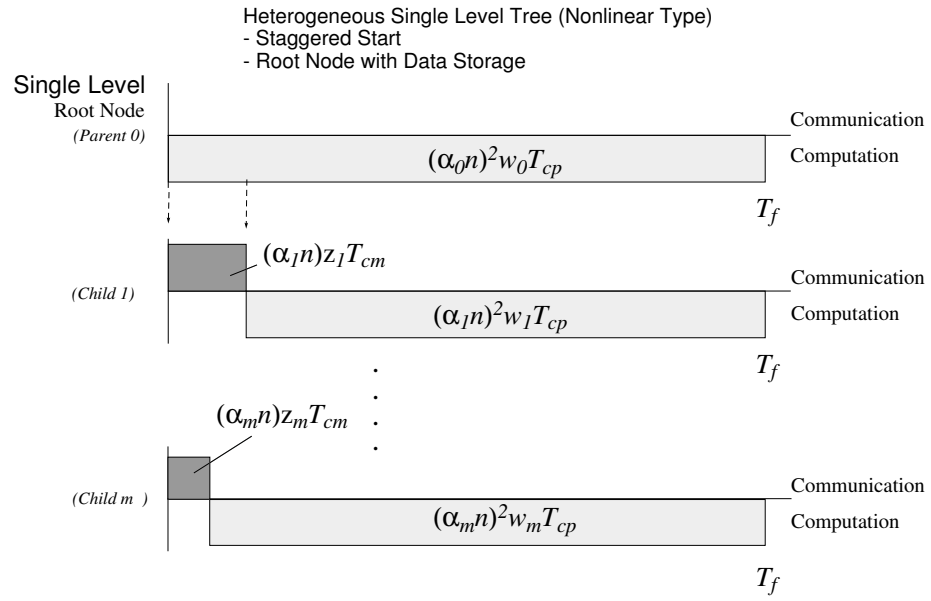

Fig. 3. Timing diagram of single level tree with simultaneous distribution, staggered start.

To calculating the speedup of a tree network, four types of equations are employed in this section, which are recursive, normalization, speedup, and constraint equations.

1) Recursive equations:

As mentioned, it is known that for an optimal solution in terms of makespan for linear problems all processors should stop at the same time [1]. Thus according to the timing diagram Fig. 3, the fundamental recursive equations of the system can be formulated as follows.

$$(\alpha_0 n)^2 w_0 T_{cp} = (\alpha_i n) z_i T_{cm} + (\alpha_i n)^2 w_i T_{cp} \qquad i = 1, 2, \ldots, m \tag{28}$$

In addition, the normalization equation for a single level tree is

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{29}$$

This yields $m + 1$ equations with $m + 1$ unknowns. Manipulating the recursive equations and normalization equation can yield the solution for the fractions of load distribution. Now from (28),

$$\alpha_i^2 + \frac{z_i T_{cm}}{n w_i T_{cp}} \alpha_i - \frac{w_0 T_{cp}}{w_i T_{cp}} \alpha_0^2 = 0 \tag{30}$$

Let

$$\xi_i = \frac{w_0 T_{cp}}{w_i T_{cp}} = \frac{w_0}{w_i} \qquad i = 1, 2, \ldots, m \tag{31}$$

and let

$$\varsigma_i = \frac{z_i T_{cm}}{n w_i T_{cp}} = \frac{\sigma_i}{n} \qquad where \qquad \sigma_i = \frac{z_i T_{cm}}{w_i T_{cp}} \qquad i = 1, 2, \ldots, m \tag{32}$$

The recursive equation (30) is transformed to

$$\alpha_i^2 + \varsigma_i \alpha_i - \xi_i \alpha_0^2 = 0 \tag{33}$$

Applying the quadratic formula to (33), one obtains

$$\alpha_i = \frac{-\varsigma_i \pm \sqrt{\varsigma_i^2 + 4\xi_i \alpha_0^2}}{2 \times 1} \tag{34}$$

Since the value of α_i is the load fraction at $node_{<i>}$, it does not make any physical sense if $\alpha_i < 0$. Hence, $\alpha_i \geq 0$ and the solution of α_i becomes

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

2) Normalization equation:

Employing equation (35), the normalization equation (29) becomes

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

Using the quadratic formula for solving Eq. (36) and then assuming that the solution of α_0 is C_0 (a specific value), One finally obtains the solution of load fractions as

$$\alpha_i = \frac{-\varsigma_i + \sqrt{\varsigma_i^2 + 4\xi_i C_0^2}}{2} \tag{37}$$

3) Speedup equation:

Now if a single level tree rooted at $node_0$ is collapsed into an equivalent node, $node_0^{eq}$, and the total load size is n, the computational time can be expressed as $(n)^2 w_0^{eq} T_{cp}$ (w_0^{eq} is the inverse computing speed of the equivalent node, $node_0^{eq}$). According to the Gantt chart-like timing diagrams, Fig. 3, the computational

time of the equivalent node (or the tree network) is equal to the computational time at the root in the tree network. That is, the finish time T_f becomes

$$T_f = (n)^2 w_0^{eq} T_{cp} = (\alpha_0 \times n)^2 w_0 T_{cp} = (C_0 \times n)^2 w_0 T_{cp} \tag{38}$$

Moreover,

$$w_0^{eq} T_{cp} = \alpha_0^2 w_0 T_{cp} = C_0^2 w_0 T_{cp} \tag{39}$$

According to Definition 1 in Section 2 (i.e. $\gamma^{eq} = w_0^{eq}/w_0$), the value of γ^{eq} can be obtained from (39) as

$$\gamma^{eq} = C_0^2 = \alpha_0^2 \tag{40}$$

In this section speedup is the ratio of job solution time at one processor to job solution time at a tree network with $m+1$ processors (see Definition 2 in Section 2.) As a result,

$$Speedup = \frac{1}{\gamma^{eq}} = \frac{1}{C_0^2} = \left(\frac{1}{\alpha_0}\right)^2 \tag{41}$$

Note that speedup is a measure of the achievable parallel processing advantage.

4) Constraints:

   a) The constraint of σ_i:

      In a simultaneous distribution protocol, it is assumed that the communication speed on $link_{<i>}$ is faster than the computing speed at $node_{<i>}$ by at least (order of magnitude) 10 times. Here $node_{<i>}$ is the node receiving all the fractional load via $link_{<i>}$. One obtains

$$\sigma_i \ll \frac{1}{10} \qquad i = 1, 2, \ldots, m \tag{42}$$

      This will guarantee that the physical characteristics of tree networks comply to our analysis model. If the communication time at some node is too slow relative to its corresponding computation time, not all nodes are needed for an optimal solution [1].

   b) The property of ς_i:

      If $\sigma_i \ll 0.1$ and n is large enough, ς_i becomes infinitesimal.

   c) Range of ξ_i:

      For isometric (balanced) rather than drastically unbalanced computing power for parallel computing,

the computing speed of each node in a tree network is specified as less than or equal to the computing speed of the child's parent by a factor of m, and greater than or equal to that of parent by a factor of $1/m$. That is,

$$\frac{1}{m} \cdot \frac{1}{w_0} \leq \frac{1}{w_i} \leq m \cdot \frac{1}{w_0} \qquad i = 1, 2, \ldots, m \tag{43}$$

Hence, the condition of a balanced computing tree network is given as follows.

$$\frac{1}{m} \leq \xi_i = \frac{w_0}{w_i} \leq m \qquad i = 1, 2, \ldots, m \tag{44}$$

4.2 Some Specific Cases

Some specific cases are discussed as follows.

1) *Link Capacity and Children Computing Speed are Homogeneous:*

Considering a homogeneous network where all children processors have the same inverse computing speed and all links have the same inverse transmission speed, then $w_i = w$ and $z_i = z$ for $i = 1, 2, \ldots, m$ (Note that the root inverse computing speed, w_0, can be different from w_i). According to (31),

$$\xi_i = \frac{w_0 T_{cp}}{w T_{cp}} = \frac{w_0}{w} = \xi \qquad i = 1, 2, \ldots, m \tag{45}$$

Now from (32) we obtains

$$\varsigma_i = \frac{z T_{cm}}{n w T_{cp}} = \varsigma = \frac{\sigma}{n} \qquad \sigma = z T_{cm} / w T_{cp} \qquad and \qquad i = 1, 2, \ldots, m \tag{46}$$

Accordingly, the constraints are specified as follows.

a) σ condition: $\sigma \ll 0.1$ for the simultaneous distribution model.

b) ς condition: If $\sigma \ll 0.1$ and n is large enough, ς becomes infinitesimal.

c) ξ condition:

$$\frac{1}{m} \leq \xi = \frac{w_0}{w} \leq m \tag{47}$$

Equation (35) becomes

$$\alpha_i = \frac{-\varsigma + \sqrt{\varsigma^2 + 4\xi\alpha_0^2}}{2} \qquad i = 1, 2, \ldots, m \tag{48}$$

Hence the normalization equation (36) becomes

$$\alpha_0 + \sum_{i=1}^{m} \frac{-\varsigma + \sqrt{\varsigma^2 + 4\xi\alpha_0^2}}{2} = 1$$
$$\alpha_0 + m \cdot \frac{-\varsigma + \sqrt{\varsigma^2 + 4\xi\alpha_0^2}}{2} = 1 \tag{49}$$

Furthermore, manipulate equation (49)

$$2\alpha_0 - m\varsigma + m\sqrt{\varsigma^2 + 4\xi\alpha_0^2} = 2 \tag{50}$$

$$(m^2\xi - 1)\alpha_0^2 + (2 + m\varsigma)\alpha_0 - (m\varsigma + 1) = 0 \tag{51}$$

Applying the quadratic formula to (51), one obtains

$$\alpha_0 = \frac{-(2 + m\varsigma) \pm \sqrt{(2 + m\varsigma)^2 + 4(m^2\xi - 1)(m\varsigma + 1)}}{2 \cdot (m^2\xi - 1)}$$
$$= \frac{-(2 + m\varsigma) \pm \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}}{2(m^2\xi - 1)} \tag{52}$$

Since α_0 is the fraction of load for computation at the root node, it does not make any physical sense if the value of α_0 is less than zero. According to the ξ condition from (47),

$$\frac{1}{m} \leq \xi \leq m \tag{53}$$

then

$$m^2 \geq m\xi \geq 1 \tag{54}$$

Because the number of children nodes is assumed to be greater than 2 in a single level tree or in subtrees of a multilevel tree, one obtains

$$m^3 \geq m^2\xi \geq m > 1 \tag{55}$$

Because the plus sign in (52) is taken instead of symbol $\pm$ and the value of α_0 is greater than zero (where $2(m^2\xi - 1) > 0$ and $-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)} > 0$), the solution of α_0 becomes

$$\alpha_0 = \frac{-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}}{2(m^2\xi - 1)} \tag{56}$$

where $\quad 4m^3\xi\varsigma + 4m^2\xi - 4 - 4m\varsigma > 0 \quad$ and $\quad 2(m^2\xi - 1) > 0.$

As a result, Eq. (40) becomes

$$\gamma^{eq} = \alpha_0^2 = \left(\frac{-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}}{2(m^2\xi - 1)} \right)^2 \tag{57}$$

and Eq. (41) becomes

$$Speedup = \frac{1}{\gamma^{eq}} = \left(\frac{2(m^2\xi - 1)}{-(2 + m\varsigma) + \sqrt{m^2\varsigma^2 + 4m^2\xi(m\varsigma + 1)}} \right)^2 \tag{58}$$

2) *Fast Communication Case:*

Let $\sigma \ll 0.1$, ς becomes infinitesimal (fast communication).

Equation (46) is repeated here as follows.

$$\varsigma = \frac{zT_{cm}}{nwT_{cp}} = \frac{\sigma}{n} \tag{59}$$

Provided that communication speed is faster than computing speed, σ is much smaller than 0.1, $\sigma \ll 0.1$. Now from Eq. (59) $m\varsigma$ is equal to $m\sigma/n$; that is $m\varsigma = m\sigma/n$. If one assume $m \ll n$, then $m\varsigma \ll 1$. Accordingly, $m\varsigma + 1$ is approaching 1 ($m\varsigma + 1 \rightarrow 1$) and $2 + m\varsigma$ is approaching 2 ($2 + m\varsigma \rightarrow 2$). The speedup formula (58) can be approximated as

$$Speedup = \left(\frac{2(m^2\xi - 1)}{-2 + \sqrt{m^2\varsigma^2 + 4m^2\xi}} \right)^2 \tag{60}$$

Because $m^2\xi > 1$ and $(m\varsigma)^2 \ll 1$, $4m^2\xi + m^2\varsigma^2$ approaches $4m^2\xi$ ($4m^2\xi + m^2\varsigma^2 \rightarrow 4m^2\xi$). Moreover, Eq. (60) becomes

$$Speedup = \left(\frac{2(m^2\xi - 1)}{-2 + \sqrt{4m^2\xi}} \right)^2 = \left(\frac{2(m^2\xi - 1)}{2(m\sqrt{\xi} - 1)} \right)^2 = \left(m\sqrt{\xi} + 1 \right)^2 \tag{61}$$

3) *Homogeneous Computing Case ($\xi = 1$):*

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

$$Speedup = (m + 1)^2 \tag{62}$$

This simple case makes intuitive sense if communication is much faster than computation. Note that speedup here is of a greater rate than that of a tree network consisting of the same number of processors but with a linear computing function in the size of the fractional load.

5 Simultaneous Distribution in a Layer Homogeneous Multilevel Fat Tree Analysis

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

In this work, store and forward switching (in contrast to cut through switching) is studied. In store and forward switching, load must be completely received by a node before being distributed to its descendants. The process of load distribution for a multilevel fat tree network using store and forward switching from upper level to lower level can be represented by a Gantt chart-like timing diagram (see Fig. 4). We will derive the speedup of the entire multilevel tree by moving upwards through the tree, collapsing successive subtrees into equivalent processors until the entire single level tree is collapsed into an equivalent node. We find that each "box" (level) in Fig. 4 illustrates the scheduling levels of a multilevel tree where the root node has data storage (all load is available at the single level tree root at time $t = 0$). The nested, shaded boxes indicate single level trees which are collapsed into equivalent nodes.

5.1 Speedup Derivation for a Multilevel Tree: Level j Subtree

Again, four type of equations are identified for calculating speedup.

1) Recursive equations:

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

$$(\alpha_{j,i}n_{j,0}^{eq})^2 w_{j-1}^{eq} T_{cp} > (\alpha_{j,i}n_{j,0}^{eq}) z_{j-1} T_{cm} \qquad i = 1, 2, \ldots, m \tag{63}$$

in this subtree (see Fig. 5), then communication time is faster than computation time. According to Fig. 5, the fundamental recursive equations of the jth level subtree network are

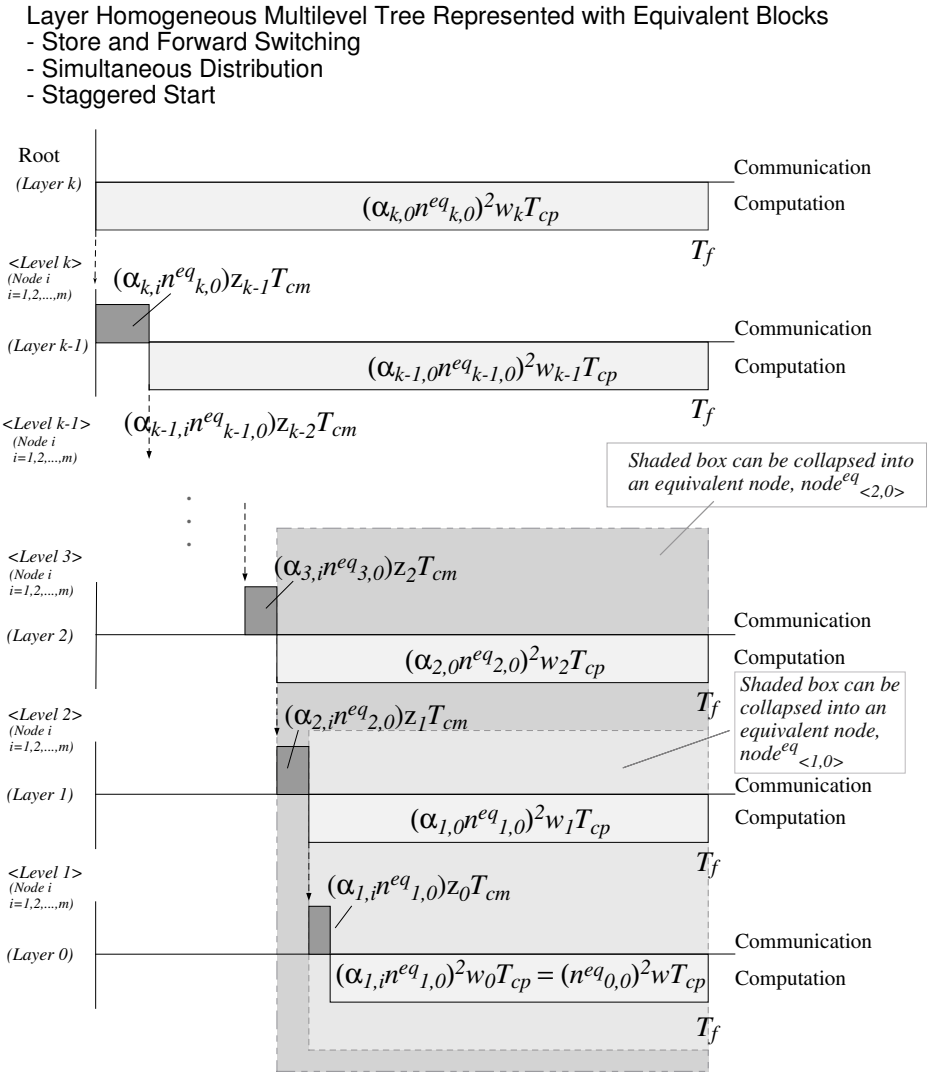

Fig. 4. Timing diagram of a layer homogeneous multilevel tree using store and forward switching, simultaneous distribution, and staggered start. The root node is with data storage. Subtrees from bottom most level to the top most level are collapsed into equivalent nodes.

$$(\alpha_{j,0}n^{eq}_{j,0})^2 w_j T_{cp} = (\alpha_{j,i}n^{eq}_{j,0})^2 w^{eq}_{j-1} T_{cp} + (\alpha_{j,i}n^{eq}_{j,0}) z_{j-1} T_{cm} \qquad j = 1, 2, \ldots, k \quad \text{and} \quad i = 1, 2, \ldots, m \tag{64}$$

The normalization equation for the jth single level subtree is

$$\alpha_{j,0} + \alpha_{j,1} + \alpha_{j,2} + \cdots + \alpha_{j,m} = 1 \tag{65}$$

This yields $m + 1$ equations with $m + 1$ unknowns. Rearranging equation (64), one has:

$$\alpha^2_{j,i} + \frac{z_{j-1}T_{cm}}{n^{eq}_{j,0} w^{eq}_{j-1} T_{cp}} \alpha_{j,i} - \frac{w_j T_{cp}}{w^{eq}_{j-1} T_{cp}} \alpha^2_{j,0} = 0 \tag{66}$$

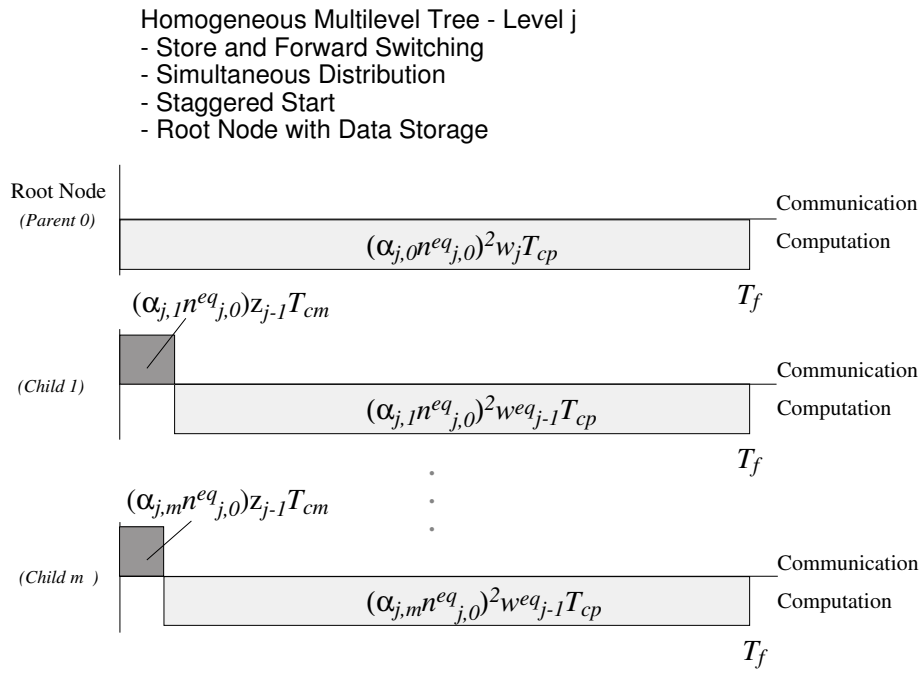

Fig. 5. Timing diagram of jth level subtree with simultaneous distribution, staggered start, and root node with data storage.

Let

$$\varsigma^{eq}_{j-1} = \frac{z_{j-1}T_{cm}}{n^{eq}_{j,0} w^{eq}_{j-1} T_{cp}} = \sigma^{eq}_{j-1}/n^{eq}_{j,0} \qquad where \quad \sigma^{eq}_{j-1} = \frac{z_{j-1}T_{cm}}{w^{eq}_{j-1}T_{cp}} \tag{67}$$

and let

$$\xi^{eq}_{j-1} = \frac{w_j}{w^{eq}_{j-1}} \tag{68}$$

According to (4), $\gamma^{eq}_j = w^{eq}_j / w_j$, we may denote

$$\sigma^{eq}_{j-1} = \frac{z_{j-1}T_{cm}}{w^{eq}_{j-1}T_{cp}} = \frac{z_{j-1}T_{cm}}{\gamma^{eq}_{j-1} w_{j-1} T_{cp}} = \frac{\sigma_{j-1}}{\gamma^{eq}_{j-1}} \qquad where \quad \sigma_{j-1} = \frac{z_{j-1}T_{cm}}{w_{j-1}T_{cp}} \tag{69}$$

and

$$\xi^{eq}_{j-1} = \frac{w_j}{w^{eq}_{j-1}} = \frac{w_j}{\gamma^{eq}_{j-1} w_{j-1}} = \frac{\xi_{j-1}}{\gamma^{eq}_{j-1}} \qquad where \quad \xi_{j-1} = \frac{w_j}{w_{j-1}} \tag{70}$$

The recursive equation (66) is transformed to

$$\alpha^2_{j,i} + \varsigma^{eq}_{j-1}\alpha_{j,i} - \xi^{eq}_{j-1}\alpha^2_{j,0} = 0 \tag{71}$$

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

$$\alpha_{j,i} = \frac{-\varsigma_{j-1}^{eq} + \sqrt{(\varsigma_{j-1}^{eq})^2 + 4\xi_{j-1}^{eq}\alpha_{j,0}^2}}{2} \qquad i = 1, 2, \ldots, m \tag{72}$$

The fraction of distribution load, $\alpha_{j,0}$, can be solved by employing the normalization equation (65).

2) Normalization equation:

According to (72), equation (65) becomes

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

$$\alpha_{j,0} + \sum_{i=1}^{m} \frac{-\varsigma_{j-1}^{eq} + \sqrt{(\varsigma_{j-1}^{eq})^2 + 4\xi_{j-1}^{eq}\alpha_{j,0}^2}}{2} = 1$$

Consequently,

$$(m^2\xi_{j-1}^{eq} - 1)\alpha_{j,0}^2 + (m\varsigma_{j-1}^{eq} + 2)\alpha_{j,0} - (m\varsigma_{j-1}^{eq} + 1) = 0$$

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

$$\alpha_{j,0} = \frac{-(m\varsigma_{j-1}^{eq} + 2) + \sqrt{(m\varsigma_{j-1}^{eq})^2 + 4m^2\xi_{j-1}^{eq}(m\varsigma_{j-1}^{eq} + 1)}}{2(m^2\xi_{j-1}^{eq} - 1)} \tag{73}$$

3) Speedup equation:

If a subtree rooted at $node_{<j,0>}$ is collapsed into an equivalent node, $node_{<j,0>}^{eq}$, the equivalent computational time at $node_{<j,0>}^{eq}$ is equal to that at $node_{<j,0>}$. If the fractional load assigned to $node_{<j,0>}^{eq}$ is $Load_{<j,0>}^{eq}$ (or $n_{j,0}^{eq}$), then the fractional load of $node_{<j,0>}$ is $\alpha_{j,0}Load_{<j,0>}^{eq}$ (or $\alpha_{j,0}n_{j,0}^{eq}$) and the fractional load of the equivalent node, $node_{<j,i>}^{eq}$ is $\alpha_{j,i}Load_{<j,0>}^{eq}$ (or $\alpha_{j,i}n_{j,0}^{eq}$). One obtains the equation

$$(1 \cdot n_{j,0}^{eq})^2 w_j^{eq} T_{cp} = (\alpha_{j,0} n_{j,0}^{eq})^2 w_j T_{cp} \tag{74}$$

Consequently, we obtain

$$\gamma_j^{eq} = \frac{w_j^{eq}}{w_j} = \alpha_{j,0}^2$$

$$Speedup = \frac{1}{\gamma_j^{eq}} \qquad j = 1, 2, \ldots, k \tag{75}$$

For a *multilevel fat tree with homogeneous layers*, the computation capability of the leaves, the nodes at the bottommost layer, can be denoted as $w_0 = w$, and w_0^{eq} is considered as w_0 at the bottommost layer. Thus, γ_0^{eq} can be obtained.

$$\gamma_0^{eq} = \frac{w_0^{eq}}{w_0} = \frac{w}{w} = 1 \tag{76}$$

4) Constraints:

a) The constraint of σ_j^{eq}:

In a simultaneous distribution protocol it is assumed that the communication speed on $link_{<j>}$ is faster than the computing speed, $1/w_j^{eq}$, at $node_{<j>}^{eq}$ at least (order of magnitude) 10 times. This will guarantee that the physical characteristics of tree networks comply to our analysis model. Note that $node_{<j>}^{eq}$ is the equivalent node receiving all the fractional load via $link_{<j>}$.

$$\sigma_j^{eq} \ll \frac{1}{10} \qquad i = 1, 2, \ldots, m \tag{77}$$

b) The property of ς_j^{eq}:

If $\sigma_j^{eq} \ll 0.1$ and n is large enough, ς_j becomes infinitesimal.

c) The range of ξ_{j-1}^{eq}:

In a tree network implemented here for parallel computing it is assumed that the computing speed of each child is less than or equal to the computing speed of the child's parent by a factor of m, and greater than or equal to that of parent by a factor of $1/m$. This constraint makes the computing capability of every node in a tree network more isometric rather than radically unbalanced. Consequently,

$$\frac{1}{m} \cdot \frac{1}{w_j} \leq \frac{1}{w_{j-1}^{eq}} \leq m \cdot \frac{1}{w_j} \qquad i = 1, 2, \ldots, m \tag{78}$$

Hence, the condition of a balanced computing tree network is given as follows.

$$\frac{1}{m} \leq \xi_{j-1}^{eq} = \frac{w_j}{w_{j-1}^{eq}} \leq m \qquad i = 1, 2, \ldots, m \tag{79}$$

5.2 Some Specific Cases

If ς_{j-1}^{eq} approaches zero, (large tree where communication is much faster than computation), the model approaches an ideal case. Each node can receive the load instantly and compute the data immediately. Under such an assumption,

the function (73) can be approximated as

$$\alpha_{j,0} = \frac{-2 + \sqrt{4m^2\xi_{j-1}^{eq}}}{2(m^2\xi_{j-1}^{eq} - 1)} = \frac{1}{m\sqrt{\xi_{j-1}^{eq}} + 1} = \frac{1}{m\sqrt{\frac{\xi_{j-1}}{\gamma_{j-1}^{eq}}} + 1} \tag{80}$$

1) *The Nodes in the Same Layer Have the Same Computing Speed:*

Let $\xi_{j-1}^{eq} = 1$, that is, the computing speed at the parent, $node_{<j,0>}$, is equal to those at its equivalent children nodes, $node_{<j,i>}^{eq}$. Thus, γ_j^{eq} becomes

$$\gamma_j^{eq} = \alpha_{j,0}^2 = \frac{1}{(m+1)^2} \qquad where \quad j = 1, 2, ..., k \tag{81}$$

and $\gamma_0^{eq} = 1$. Consequently, one obtains speedup for a single level tree network as follows.

$$Speedup = (m+1)^2 \tag{82}$$

Now from equation (4) and equation (70),

$$\gamma_{j-1}^{eq} = \xi_{j-1} = \frac{w_j}{w_{j-1}} = \frac{w_{j-1}^{eq}}{w_{j-1}} \tag{83}$$

Let $w_0 = w$, then $w_1 = \gamma_0^{eq} w_0 = 1 \cdot w = w$. Therefore, one obtains the inverse computing speed for each layer node as follows.

$$\begin{aligned} w_j &= \gamma_{j-1}^{eq} \times w_{j-1} = \frac{1}{(m+1)^2} \times w_{j-1} \\ &= \left[\frac{1}{(m+1)^2}\right]^{j-1} \times w_1 = \left[\frac{1}{(m+1)^2}\right]^{j-1} \times w \qquad where \quad j = 2, 3, ..., k \end{aligned} \tag{84}$$

2) *All Nodes Have the Same Computing Speed:*

Let $\xi_{j-1} = 1$, that is, the inverse computing speed of all nodes is the same and designated as w, then γ_j^{eq} becomes

$$\gamma_j^{eq} = \alpha_{j,0}^2 = \frac{1}{\left(\frac{m}{\sqrt{\gamma_{j-1}^{eq}}} + 1\right)^2} \tag{85}$$

Because $\gamma_0 = 1$, the recurrence equation, (85), is induced as follows.

$$\gamma_1^{eq} = \frac{1}{(m+1)^2} \tag{86}$$

$$\gamma_2^{eq} = \frac{1}{\left(\frac{m}{\sqrt{\frac{1}{(m+1)^2}}} + 1\right)^2} = \frac{1}{(m^2+m+1)^2} \tag{87}$$

$$\vdots$$

$$\gamma_j^{eq} = \frac{1}{(m^j + m^{j-1} + \cdots + m + 1)^2} = \frac{1}{(\sum_{l=0}^{j} m^l)^2} \qquad j = 1, 2, \ldots, k \tag{88}$$

Consequently, the speedup of the homogeneous multilevel tree network is

$$Speedup = \frac{1}{\gamma_k} = (\sum_{l=0}^{k} m^l)^2 \tag{89}$$

We conclude that the speedup is the square of the total number of nodes, which makes intuitive sense. Note that this speedup expression is greater than linear in the number of nodes (processors).

6 Speedup of a Single Level Tree with Sequential Distribution and with Staggered Start

Sequential load distribution is employed in this section in a heterogeneous single level tree using staggered start. It is used as the model in most of the divisible load scheduling literature. Even though a closed form solution for optimal load allocation and speedup is not possible, an iterative solution is developed.

6.1 Speedup Derivation for A Single Level Tree with Running Time $\Theta(n_i^{\chi})$

The structure of a single level tree network with root, $m + 1$ processors and m links is illustrated in Fig. 1. In this section we assume that the worst-case running cost of an algorithm is $\Theta(n_i^{\chi})$ ($i = 0, 1, 2, \ldots, m$), then the computation time function at a node becomes a power χ function in load size n_i. Still, the communication time function on a link is a linear function in its assigned load size.

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. 6. It is assumed that all of the load is available at the root node at time $t = 0$.

Four types of equations are again needed to determine the speedup. They are recursive, normalization, constraints, and speedup equations.

1) Recursive equations and normalization equation:

   According to the timing diagram Fig. 6, the fundamental recursive equations of the system can be formulated as follows:

$$(\alpha_i n)^{\chi} w_i T_{cp} = (\alpha_{i+1} n)^{\chi} w_{i+1} T_{cp} + (\alpha_{i+1} n) z_{i+1} T_{cm} \qquad i = 0, 1, 2, \ldots, m-1 \tag{90}$$

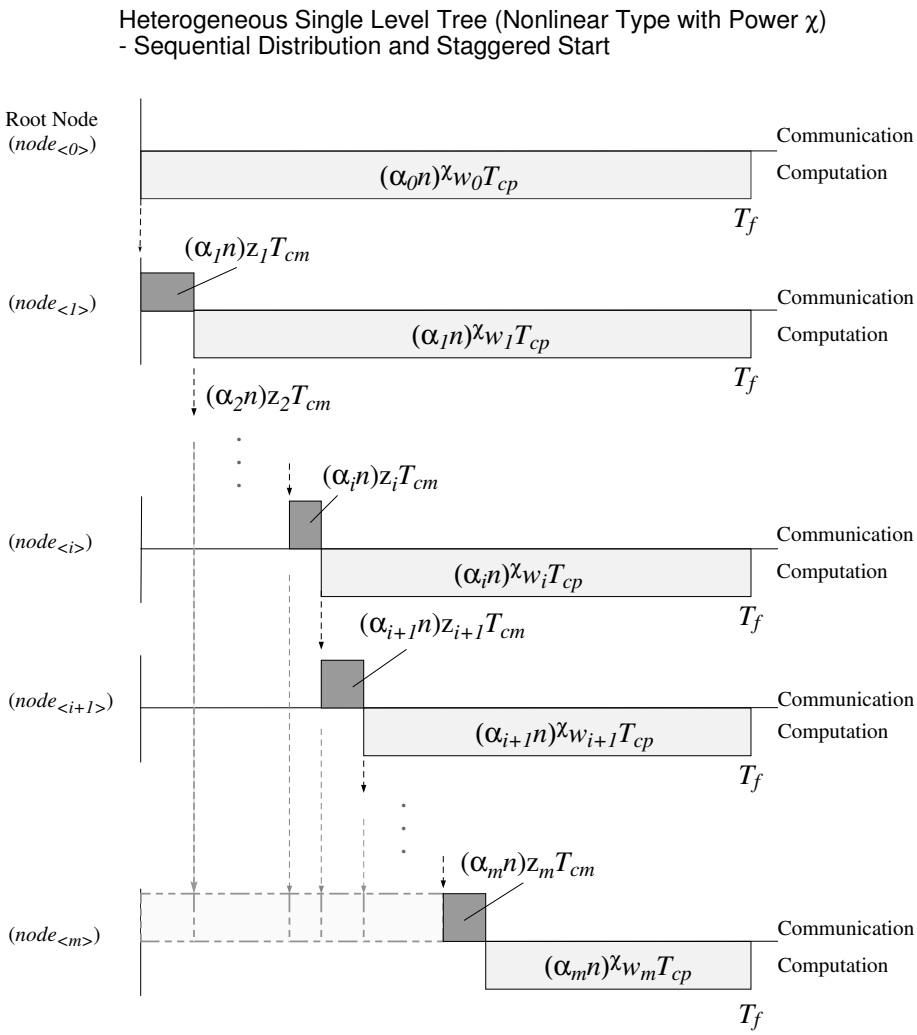

Fig. 6. Timing diagram of a heterogeneous single level tree using sequential distribution and staggered start.

The normalization equation is

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{91}$$

This yields $m+1$ equations with $m+1$ unknowns. Manipulating the recursive equations and normalization equation can yield the solution for the fractions of load distribution. Now from (90),

$$(\alpha_i n)^{\chi} = (\alpha_{i+1} n)^{\chi} \frac{w_{i+1} T_{cp}}{w_i T_{cp}} + (\alpha_{i+1} n) \frac{z_{i+1} T_{cm}}{w_i T_{cp}} \qquad i = 0, 1, 2, \ldots, m-1 \tag{92}$$

Let

$$\varsigma_i = \frac{z_i T_{cm}}{n^{\chi-1} w_i T_{cp}} = \frac{\sigma_i}{n^{\chi-1}} \tag{93}$$

where

$$\sigma_i = \frac{z_i T_{cm}}{w_i T_{cp}} \qquad i = 1, 2, \ldots, m \tag{94}$$

and let

$$\xi_{i+1} = \frac{w_{i+1} T_{cp}}{w_i T_{cp}} = \frac{w_{i+1}}{w_i} \qquad i = 1, 2, \ldots, m \tag{95}$$

Then (90) becomes

$$(\alpha_i)^\chi = \xi_{i+1}(\alpha_{i+1})^\chi + \xi_{i+1}\varsigma_{i+1}\alpha_{i+1} \qquad i = 0, 1, 2, \ldots, m-1 \tag{96}$$

2) Constraints:

a) The constraint of σ_i:

In a simultaneous distribution protocol, it is assumed that the communication speed on $link_{<i>}$ is faster than the computing speed at $node_{<i>}$ at least (order of magnitude) 10 times. This will guarantee that the physical characteristics of tree networks comply to our analysis model. Note that $node_{<i>}$ is the node receiving all the fractional load via $link_{<i>}$.

$$\sigma_i \ll \frac{1}{10} \qquad i = 1, 2, \ldots, m \tag{97}$$

b) The constraint of ς_i:

If $\sigma_i \ll 0.1$ and n is large enough, ς_i becomes infinitesimal.

c) Range of ξ_i:

In a tree network considered here for parallel computing it is assumed that the computing speed of each child is less than or equal to the computing speed of the child's parent by a factor of m, and greater than or equal to that of parent by a factor of $1/m$. This constraint makes the computing capability of every node in a tree network more isometric rather than radically unbalanced. Consequently,

$$\frac{1}{m} \cdot \frac{1}{w_0} \leq \frac{1}{w_i} \leq m \cdot \frac{1}{w_0} \qquad i = 1, 2, \ldots, m \tag{98}$$

The condition of a balanced computing tree network is given as follows.

$$\frac{1}{m} \leq x_i = \frac{w_0}{w_i} \leq m \qquad i = 1, 2, \ldots, m \tag{99}$$

The matrix equation consists of recursive equations and normalization equation are represented as follows.

$$\begin{bmatrix} \alpha_0^{\chi} \\ \alpha_1^{\chi} \\ \alpha_2^{\chi} \\ \alpha_3^{\chi} \\ \vdots \\ \alpha_{m-1}^{\chi} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & \xi_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \xi_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \xi_3 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & \xi_m \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0^{\chi} \\ \alpha_1^{\chi} \\ \alpha_2^{\chi} \\ \alpha_3^{\chi} \\ \vdots \\ \alpha_{m-1}^{\chi} \\ \alpha_m^{\chi} \end{bmatrix} + \begin{bmatrix} 0 & \xi_1\varsigma_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \xi_2\varsigma_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \xi_3\varsigma_3 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & \xi_m\varsigma_m \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{m-1} \\ \alpha_m \end{bmatrix}$$

These unknowns, $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_m$, can be solved by standard iterative techniques. That is, one substitutes an initial guess of the α (and α^{χ}) vector into the right hand side of the matrix equation, to create the (left hand side) new estimate of the α^{χ} vector which is then substituted into the right side, an on and on, until convergence occurs.

3) Alternative recursive equations and normalization equation:

According to the timing diagram Fig. 6, the fundamental recursive equations of the system can be formulated as follows:

$$(\alpha_0 n)^{\chi} w_0 T_{cp} = (\alpha_i n)^{\chi} w_i T_{cp} + \sum_{h=1}^{i} (\alpha_h n) z_h T_{cm} \qquad i = 1, 2, \dots, m \tag{100}$$

The normalization equation is

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{101}$$

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

Equation Eq. (100) becomes

$$(\alpha_i)^{\chi} w_i + \sum_{h=1}^{i} \alpha_h \varsigma_h w_h = (\alpha_0)^{\chi} w_0 \qquad i = 1, 2, \dots, m \tag{102}$$

where

$$\varsigma_h = \frac{z_h T_{cm}}{n^{\chi-1} w_h T_{cp}} = \frac{\sigma_h}{n^{\chi-1}} \tag{103}$$

The matrix equation consists of recursive equations and normalization equation are represented as follows.

$$\begin{bmatrix} 1 \\ \alpha_0^{\chi} w_0 \\ \alpha_0^{\chi} w_0 \\ \alpha_0^{\chi} w_0 \\ \vdots \\ \alpha_0^{\chi} w_0 \\ \alpha_0^{\chi} w_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha_1^{\chi} w_1 \\ \alpha_2^{\chi} w_2 \\ \alpha_3^{\chi} w_3 \\ \vdots \\ \alpha_{m-1}^{\chi} w_{m-1} \\ \alpha_m^{\chi} w_m \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 \\ 0 & \varsigma_1 w_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \varsigma_1 w_1 & \varsigma_2 w_2 & 0 & \dots & 0 & 0 \\ 0 & \varsigma_1 w_1 & \varsigma_2 w_2 & \varsigma_3 w_3 & \dots & 0 & 0 \\ 0 & \varsigma_1 w_1 & \varsigma_2 w_2 & \varsigma_3 w_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \varsigma_1 w_1 & \varsigma_2 w_2 & \varsigma_3 w_3 & \dots & \varsigma_{m-1} w_{m-1} & \varsigma_m w_m \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{m-1} \\ \alpha_m \end{bmatrix}$$

These unknowns, $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_m$, can, again, be solved iteratively.

4) Speedup equation:

Now, if a single level tree rooted at $node_0$ is collapsed into an equivalent node, $node_0^{eq}$, and the total load size is n, the computational time can be expressed as $(n)^{\chi} w_0^{eq} T_{cp}$ (w_0^{eq} is the inverse computing speed of the equivalent node, $node_0^{eq}$). According to the Gantt chart-like timing diagrams, Fig. 6, the computational time of the equivalent node (or the tree network) is equal to the computational time at the root in the tree network. Consequently, the finish time T_f becomes

$$T_f = (n)^{\chi} w_0^{eq} T_{cp} = (\alpha_0 \times n)^{\chi} w_0 T_{cp} \tag{104}$$

Hence,

$$w_0^{eq} T_{cp} = \alpha_0^{\chi} w_0 T_{cp} \tag{105}$$

According to Definition 1 in Section 2 (i.e. $\gamma^{eq} = w_0^{eq}/w_0$) and Eq. (105), the value of γ^{eq} becomes

$$\gamma^{eq} = \alpha_0^{\chi} \tag{106}$$

Thus,

$$Speedup = \frac{1}{\gamma^{eq}} = \left(\frac{1}{\alpha_0}\right)^{\chi} \tag{107}$$

7 Conclusion and Lessons Learned

A number of findings have resulted from this study:

• It is possible to solve for optimal load allocations and speedup for models with nonlinear power law computational complexity, either in closed form or iteratively. A proof has been provided of the condition for optimal load distribution of nonlinear loads.

• Nonlinear problems have a need for post-processing, because of the dependency of the input data when processed by a nonlinear algorithm.

• We corroborate the results of Drozdowksi and Wolniewicz [4] that super-linear speedup can result for nonlinear divisible load processing.

• It should be pointed out that higher order nonlinear equations can suffer from numerical error (due to finite computer word size) problems and so some care is warranted.

We have sought to show how to demonstrate the possibility of optimal scheduling for a number of representative scheduling policies on tree interconnection networks under power law nonlinearties in the space available. Of course for specific applications other scheduling policies, nonlinear functional forms and topologies may be of interest. Because of the super-linear speedup, parallel processing of loads with nonlinear computational complexity is a promising technique to maximize computational efficiency on multiple processor systems.

Appendix

The following theorem is proved in this appendix [1].

Thrm: *Given that load distribution in a single level tree follows the optimal sequencing condition, then if all the nodes of the nonlinear computing model receiving nonzero load fractions stop computing at the same time, the processing time (makespan) is minimum for the specific scheduling strategies indicated.*

The optimal sequencing condition means that the sequence of load distribution used by the root node should follow the order in which the link speeds decrease. That is, the relationship of the inverse link speeds, z_i, is

$$z_1 < z_2 < \cdots < z_i < \cdots < z_m \qquad i = 1, 2, \ldots, m \tag{108}$$

In this Appendix two types of distribution in single level trees are taken into account, sequential distribution (see Fig. 6) and simultaneous distribution (See Fig. 3). For sequential distribution the optimal sequencing condition is required to prevent some nodes from being assigned zero fractions, but it can be relaxed for simultaneous distribution.

In a single level tree, we assume that there are $m+1$ nodes ($node_{<0>}, node_{<1>}, \ldots, node_{<m>}$), and m links (l_1, …, l_m). Before the proof, some definitions are first illustrated as follows [1]:

1) *Load distribution:* $\boldsymbol{\alpha}$ is an ordered $m+1$ tuple.

$$\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m) \tag{109}$$

where α_i is the load fraction assigned to $node_{<i>}$. Further, the normalization equation is

$$\sum_{i=0}^{m} \alpha_i = 1 \qquad where \qquad 0 \le \alpha_i \le 1 \qquad i = 0, 1, \ldots, m \tag{110}$$

The set of all feasible load distributions is denoted by L.

2) *Finish time:* The finish time of $node_{<i>}$ is denoted by $T_i(\boldsymbol{\alpha})$, for a given load distribution $\boldsymbol{\alpha} \in L$.

3) *Processing time:* For a given $\boldsymbol{\alpha} \in L$, this is defined as

$$T(\boldsymbol{\alpha}) = \max\{T_0(\boldsymbol{\alpha}), T_1(\boldsymbol{\alpha}), \ldots, T_m(\boldsymbol{\alpha})\} \tag{111}$$

In other words, $T(\boldsymbol{\alpha})$ is the time at which the entire load is processed.

4) *Minimum processing time:* This is defined as

$$T^* = \min_{\boldsymbol{\alpha} \in L} T(\boldsymbol{\alpha}) \tag{112}$$

5) *Optimal load distribution:* This is defined as the load distribution $\boldsymbol{\alpha}^* \in L$ such that the processing time is a minimum, that is,

$$\boldsymbol{\alpha}^* = arg \min_{\boldsymbol{\alpha} \in L} T(\boldsymbol{\alpha}) \tag{113}$$

Subsequently, we will prove both the sequential distribution and simultaneous distribution cases by the contradiction method.

I. Sequential Distribution: (See Fig. 6)

Proof: Let $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m) \in L$ be the initial load distribution such that all the nodes stop computing at the same time. Provided that the processing time is not a minimum, there must exist an $\boldsymbol{\alpha^*} = (\alpha_0^*, \alpha_1^*, \alpha_2^*, \ldots, \alpha_m^*) \in L$ such that $\boldsymbol{\alpha^*}$ satisfiies

$$\boldsymbol{\alpha^*} = arg \min_{\boldsymbol{\alpha} \in L} T(\boldsymbol{\alpha}) \tag{114}$$

As a consequence,

$$T_j(\boldsymbol{\alpha^*}) < T_j(\boldsymbol{\alpha}) \qquad where \qquad j = 0, 1, 2, \ldots, m \tag{115}$$

Our approach in this proof is to show that $\alpha_i^* < \alpha_i$ for all i but this contradicts a normalization condition. We note that the case where $T_j(\boldsymbol{\alpha^*}) \leq T_j(\boldsymbol{\alpha})$ can be handled with minor modifications. Because the finish time of the root $node_{<0>}$ is $(\alpha_0 n)^{\chi} w_0 T_{cp}$, equation (115), $T_0(\boldsymbol{\alpha^*}) < T_0(\boldsymbol{\alpha})$, becomes

$$(\alpha_0^* n)^{\chi} w_0 T_{cp} < (\alpha_0 n)^{\chi} w_0 T_{cp} \tag{116}$$

From (100) the finish time of child node, $node_{<i>}$, is $(\alpha_i n)^{\chi} w_i T_{cp} + \sum_{h=1}^{i} (\alpha_h n) z_h T_{cm}$. According to equation (115), $T_i(\boldsymbol{\alpha^*}) < T_i(\boldsymbol{\alpha})$ (where $i = 1, 2, \ldots, m$), one obtains

$$(\alpha_i^* n)^{\chi} w_i T_{cp} + \sum_{h=1}^{i} (\alpha_h^* n) z_h T_{cm} < (\alpha_i n)^{\chi} w_i T_{cp} + \sum_{h=1}^{i} (\alpha_h n) z_h T_{cm} \qquad i = 1, 2, \ldots, m \tag{117}$$

Let χ be an integer and $\chi \geq 1$. Equation (116) becomes

$$((\alpha_0^*)^{\chi} - (\alpha_0)^{\chi}) n^{\chi} w_0 T_{cp} < 0 \tag{118}$$

Hence,

$$(\alpha_0^* - \alpha_0) \left\{ (\alpha_0^*)^{\chi-1} + (\alpha_0^*)^{\chi-2} \alpha_0 + \cdots + (\alpha_0^*)^1 (\alpha_0)^{\chi-2} + (\alpha_0)^{\chi-1} \right\} n^{\chi} w_0 T_{cp} < 0 \tag{119}$$

Because α_i^*, α_i, n, w_0, and T_{cp} are all positive, then

$$\left\{ (\alpha_0^*)^{\chi-1} + (\alpha_0^*)^{\chi-2} \alpha_0 + \cdots + (\alpha_0^*)^1 (\alpha_0)^{\chi-2} + (\alpha_0)^{\chi-1} \right\} n^{\chi} w_0 T_{cp} > 0 \tag{120}$$

Consequently,

$$(\alpha_0^* - \alpha_0) < 0 \tag{121}$$

$$\alpha_0^* < \alpha_0 \tag{122}$$

At child $node_{<m>}$, one obtains finish time $T_m(\boldsymbol{\alpha^*}) < T_m(\boldsymbol{\alpha})$, then

$$(\alpha_m^* n)^\chi w_m T_{cp} + \sum_{h=1}^{m} (\alpha_h^* n) z_h T_{cm} < (\alpha_m n)^\chi w_m T_{cp} + \sum_{h=1}^{m} (\alpha_h n) z_h T_{cm}$$

$$((\alpha_m^*)^\chi - (\alpha_m)^\chi) n^\chi w_m T_{cp} < \sum_{h=1}^{m} (\alpha_h - \alpha_h^*) n z_h T_{cm} \tag{123}$$

According to (108)

$$\sum_{h=1}^{m} (\alpha_h - \alpha_h^*) n z_h T_{cm} < \sum_{h=1}^{m} (\alpha_h - \alpha_h^*) n z_m T_{cm} \tag{124}$$

According to the normalization equations,

$$\sum_{h=0}^{m} \alpha_h^* = 1 \qquad and \qquad \sum_{h=0}^{m} \alpha_h = 1 \tag{125}$$

That is,

$$\sum_{h=1}^{m} \alpha_h^* = 1 - \alpha_0^* \qquad and \qquad \sum_{h=1}^{m} \alpha_h = 1 - \alpha_0 \tag{126}$$

Therefore, from equation (123), (124), and (126)

$$((\alpha_m^*)^\chi - (\alpha_m)^\chi) n^\chi w_m T_{cp} < (1 - \alpha_0) n z_m T_{cm} - (1 - \alpha_0^*) n z_m T_{cm} \tag{127}$$

According to (121)

$$(1 - \alpha_0) n z_m T_{cm} - (1 - \alpha_0^*) n z_m T_{cm} = (\alpha_0^* - \alpha_0) n z_m T_{cm} < 0 \tag{128}$$

Hence,

$$((\alpha_m^*)^\chi - (\alpha_m)^\chi) n^\chi w_m T_{cp} < 0 \tag{129}$$

$$\alpha_m^* < \alpha_m \tag{130}$$

Similarly, $T_{m-1}(\alpha^*) < T_{m-1}(\alpha^*)$ at child $node_{<m-1>}$. This leads to

$$(\alpha_{m-1}^* n)^\chi w_{m-1} T_{cp} + \sum_{h=1}^{m-1} (\alpha_h^* n) z_h T_{cm} < (\alpha_{m-1} n)^\chi w_{m-1} T_{cp} + \sum_{h=1}^{m-1} (\alpha_h n) z_h T_{cm}$$

$$((\alpha_{m-1}^*)^\chi - (\alpha_{m-1})^\chi) n^\chi w_{m-1} T_{cp} < \sum_{h=1}^{m-1} (\alpha_h - \alpha_h^*) n z_h T_{cm} \tag{131}$$

According to (108),

$$\sum_{h=1}^{m-1}(\alpha_h-\alpha_h^*)nz_hT_{cm}<\sum_{h=1}^{m-1}(\alpha_h-\alpha_h^*)nz_{m-1}T_{cm} \tag{132}$$

According to the normalization equations,

$$\sum_{h=0}^{m}\alpha_h^*=1 \qquad and \qquad \sum_{h=0}^{m}\alpha_h=1 \tag{133}$$

That is,

$$\sum_{h=1}^{m-1}\alpha_h^*=1-\alpha_0^*-\alpha_m^* \qquad and \qquad \sum_{h=1}^{m-1}\alpha_h=1-\alpha_0-\alpha_m \tag{134}$$

Therefore, from (131), (132), and (134),

$$((\alpha_{m-1}^*)^\chi-(\alpha_{m-1})^\chi)n^\chi w_{m-1}T_{cp}<(1-\alpha_0-\alpha_m)nz_{m-1}T_{cm}-(1-\alpha_0^*-\alpha_m^*)nz_{m-1}T_{cm} \tag{135}$$

Furthermore,

$$\begin{aligned}&(1-\alpha_0-\alpha_m)nz_{m-1}T_{cm}-(1-\alpha_0^*-\alpha_m^*)nz_{m-1}T_{cm}\\&=\{(\alpha_0^*+\alpha_m^*)-(\alpha_0+\alpha_m)\}nz_{m-1}T_{cm}<0\end{aligned} \tag{136}$$

Therefore, according to (122) and (130)

$$((\alpha_{m-1}^*)^\chi-(\alpha_{m-1})^\chi)n^\chi w_{m-1}T_{cp}<0 \tag{137}$$

$$\alpha_{m-1}^*<\alpha_{m-1} \tag{138}$$

As a result, at $node_{<i>}$, $T_i(\boldsymbol{\alpha^*})<T_i(\boldsymbol{\alpha})$ (where $i=1,2,\ldots,m$),

$$\begin{aligned}&(\alpha_i^*n)^\chi w_iT_{cp}+\sum_{h=1}^{i}(\alpha_h^*n)z_hT_{cm}<(\alpha_in)^\chi w_iT_{cp}+\sum_{h=1}^{i}(\alpha_hn)z_hT_{cm}\\&((\alpha_i^*)^\chi-(\alpha_i)^\chi)n^\chi w_iT_{cp}<\sum_{h=1}^{i}(\alpha_h-\alpha_h^*)nz_hT_{cm} \qquad i=1,2,\ldots,m\end{aligned} \tag{139}$$

According to (108)

$$\sum_{h=1}^{i}(\alpha_h-\alpha_h^*)nz_hT_{cm}<\sum_{h=1}^{i}(\alpha_h-\alpha_h^*)nz_iT_{cm} \qquad i=1,2,\ldots,m \tag{140}$$

Provided that we have already obtained the following inequalities.

$$\alpha_0^*<\alpha_0;\quad \alpha_m^*<\alpha_m;\quad \alpha_{m-1}^*<\alpha_{m-1};\quad \cdots;\quad \alpha_{i+1}^*<\alpha_{i+1} \tag{141}$$

According to the normalization equations,

$$\sum_{h=0}^{m} \alpha_h^* = 1 \qquad and \qquad \sum_{h=0}^{m} \alpha_h = 1 \tag{142}$$

That is,

$$\sum_{h=1}^{i} \alpha_h^* = 1 - \alpha_0^* - \alpha_m^* - \alpha_{m-1}^* - \cdots - \alpha_{i+1}^* \tag{143}$$

$$\sum_{h=1}^{i} \alpha_h = 1 - \alpha_0 - \alpha_m - \alpha_{m-1} - \cdots - \alpha_{i+1} \tag{144}$$

Therefore, from equation (139), (140), (143), and (144)

$$\begin{aligned} &((\alpha_i^*)^\chi - (\alpha_i)^\chi) n^\chi w_i T_{cp} \\ &< (1 - \alpha_0 - \alpha_m - \alpha_{m-1} - \cdots - \alpha_{i+1}) n z_i T_{cm} - (1 - \alpha_0^* - \alpha_m^* - \cdots - \alpha_{i+1}^*) n z_i T_{cm} \end{aligned} \tag{145}$$

According to (141)

$$\begin{aligned} &(1 - \alpha_0 - \alpha_m - \alpha_{m-1} - \cdots - \alpha_{i+1}) n z_h T_{cm} - (1 - \alpha_0^* - \alpha_m^* - \cdots - \alpha_{i+1}^*) n z_h T_{cm} \\ &= \left\{ (\alpha_0^* + \alpha_m^* + \cdots + \alpha_{i+1}^*) - (\alpha_0 + \alpha_m + \alpha_{m-1} + \cdots + \alpha_{i+1}) \right\} n z_h T_{cm} < 0 \end{aligned} \tag{146}$$

Hence, from (145)

$$((\alpha_i^*)^\chi - (\alpha_i)^\chi) n^\chi w_i T_{cp} < 0 \tag{147}$$

Therefore,

$$\alpha_i^* < \alpha_i \qquad for \qquad i = 1, 2, \ldots, m \tag{148}$$

Consequently, from (122) and (148),

$$\sum_{j=0}^{m} \alpha_j^* < \sum_{j=0}^{m} \alpha_j \tag{149}$$

This leads to a contradiction since both $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha^*} \in L$ and their component should add up to one. □

II. Simultaneous Distribution: (See Fig. 3)

We assume that the nonlinear load computing in the single level tree of Fig. 3 is of power χ and $\chi \geq 1$.

The proof for the simultaneous distribution is illustrated as follows.

Proof: Let $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m) \in L$ be the initial load distribution such that all the nodes stop computing at the same time. Provided that the processing time is not a minimum, there must exist an $\boldsymbol{\alpha^*} = (\alpha_0^*, \alpha_1^*, \alpha_2^*, \ldots, \alpha_m^*) \in L$ such that $\boldsymbol{\alpha^*}$ satisfiies

$$\boldsymbol{\alpha^*} = arg \min_{\boldsymbol{\alpha} \in L} T(\boldsymbol{\alpha}) \tag{150}$$

As a consequence,

$$T_j(\boldsymbol{\alpha^*}) < T_j(\boldsymbol{\alpha}) \qquad where \qquad j = 0, 1, 2, \ldots, m \tag{151}$$

Because the finish time at the root, $node_{<0>}$, is $(\alpha_0 n)^{\chi} w_0 T_{cp}$, equation (151), $T_0(\boldsymbol{\alpha^*}) < T_0(\boldsymbol{\alpha})$, leads to

$$(\alpha_0^* n)^{\chi} w_0 T_{cp} < (\alpha_0 n)^{\chi} w_0 T_{cp} \tag{152}$$

According to (28) with power, χ, the finish time of the child node, $node_{<i>}$, becomes $(\alpha_i n)^{\chi} w_i T_{cp} + (\alpha_i n) z_i T_{cm}$. According to equation (151), $T_i(\boldsymbol{\alpha^*}) < T_i(\boldsymbol{\alpha})$ (where $i = 1, 2, \ldots, m$), one obtains

$$(\alpha_i^* n)^{\chi} w_i T_{cp} + (\alpha_i^* n) z_i T_{cm} < (\alpha_i n)^{\chi} w_i T_{cp} + (\alpha_i n) z_i T_{cm} \qquad i = 1, 2, \ldots, m \tag{153}$$

Without loss of generality, let χ be an integer and $\chi \geq 1$. Equation (152) becomes

$$((\alpha_0^*)^{\chi} - (\alpha_0)^{\chi}) n^{\chi} w_0 T_{cp} < 0 \tag{154}$$

$$(\alpha_0^* - \alpha_0) \left\{ (\alpha_0^*)^{\chi - 1} + (\alpha_0^*)^{\chi - 2} \alpha_0 + \cdots + (\alpha_0^*)^1 (\alpha_0)^{\chi - 2} + (\alpha_0)^{\chi - 1} \right\} n^{\chi} w_0 T_{cp} < 0 \tag{155}$$

Because α_i^*, α_i, n, w_0, and T_{cp} are all positive, this leads to

$$\left\{ (\alpha_0^*)^{\chi - 1} + (\alpha_0^*)^{\chi - 2} \alpha_0 + \cdots + (\alpha_0^*)^1 (\alpha_0)^{\chi - 2} + (\alpha_0)^{\chi - 1} \right\} n^{\chi} w_0 T_{cp} > 0 \tag{156}$$

Consequently, one obtains

$$(\alpha_0^* - \alpha_0) < 0$$

$$\alpha_0^* < \alpha_0 \tag{157}$$

At child $node_{<i>}$, one obtains finish time $T_i(\boldsymbol{\alpha^*}) < T_i(\boldsymbol{\alpha})$; therefore,

$$(\alpha_i^* n)^{\chi} w_i T_{cp} + (\alpha_i^* n) z_i T_{cm} < (\alpha_i n)^{\chi} w_i T_{cp} + (\alpha_i n) z_i T_{cm} \qquad i = 1, 2, \ldots, m \tag{158}$$

$$((\alpha_i^*)^{\chi} - (\alpha_i)^{\chi}) n^{\chi} w_i T_{cp} + (\alpha_i^* - \alpha_i) n z_i T_{cm} < 0 \tag{159}$$

Moreover,

$$(\alpha_i^* - \alpha_i) \left\{ \left[(\alpha_i^*)^{\chi-1} + (\alpha_i^*)^{\chi-2}\alpha_i + \cdots + (\alpha_i^*)^1(\alpha_i)^{\chi-2} + (\alpha_i)^{\chi-1} \right] n^\chi w_i T_{cp} + n z_i T_{cm} \right\} < 0 \quad (160)$$

Because α_i^*, $\alpha_i, n, w_i, z_i, T_{cp}$, and T_{cm} are all positive,

$$\left\{ \left[(\alpha_i^*)^{\chi-1} + (\alpha_i^*)^{\chi-2}\alpha_i + \cdots + (\alpha_i^*)^1(\alpha_i)^{\chi-2} + (\alpha_i)^{\chi-1} \right] n^\chi w_i T_{cp} + n z_i T_{cm} \right\} > 0 \quad (161)$$

Consequently,

$$(\alpha_i^* - \alpha_i) < 0$$

$$\alpha_i^* < \alpha_i \qquad where \qquad i = 1, 2, \ldots, m \quad (162)$$

One obtains

$$\sum_{j=0}^{m} \alpha_j^* < \sum_{j=0}^{m} \alpha_j \quad (163)$$

This leads to a contradiction since both $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha^*} \in L$ and their components should sum to one. □

REFERENCES

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

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

[3] J. T. Hung and T. G. Robertazzi, "Scalable scheduling for clusters and grids using cut through switching," *Special Issue on Cluster/Grid Computing, International Journal of Computers and their Applications*, , no. 26, pp. 147–156, March 2004.

[4] M. Drozdowksi and P. Wolniewicz, "Out-of-core divisible load processing," *IEEE Tran. Parallel and Distributed Systems*, vol. 14, pp. 1048–1056, Octber 2003.

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

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

[7] 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.

[8] 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.

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

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

[11] 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.

[12] H. J. Kim, "A novel load distribution algorithm for divisible loads," *in special issue of Cluster Computing on Divisible Load Scheduling*, vol. 6, pp. 41–46, January 2003, Issue 1.

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

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

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

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

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

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

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

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

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

[22] J.T. Hung and T.G. Robertazzi, "Scheduling nonlinear computational loads," Tech. Rep. 810, Stony Brook University College of Engineering and Applied Science, March 2004.

[23] J. T. Hung and T. G. Robertazzi, "Divisible load cut through switching in sequential tree networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, pp. 968–982, March 2004.

[24] T.G. Robertazzi, "Processor equivalence for a linear daisy chain of load sharing processors.," *IEEE Transaction on Aerospace and Electronic Systems*, vol. 29, pp. 1216–1221, 1993.

[25] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, McGraw-Hill Book Company, New York, 1998.