Scheduling Nonlinear Divisible Loads in a Single Level Tree Network

S. Suresh[1], H. J. Kim[2], Cui run[2], and T. G. Robertazzi[3]

[1] School of Computer Engineering
Nanyang Technological University
Singapore. Email: ssundaram@ntu.edu.sg

[2] Division of Information Management and Security
Korea University
Seoul, South Korea.

[3] State University of New York
Stony Brook, Stony Brook, New York, USA

July 19, 2011

Abstract: In this paper, we study the scheduling problem for polynomial time complexity computational loads in a single level tree network with collective communication model. The problem of minimizing the processing time is investigated when the computational loads require polynomial order of processing time which is proportional to the size of load fraction. In divisible load theory framework, the presence of polynomial time complexity computational loads leads to solving higher-order algebraic equations to find the optimal load fractions assigned to the processors in the network. The problem of finding optimal load fraction is a computationally intensive task. Using a mild assumption on the ratio of communication time to computation time, we present a closed-form solution for near optimal load fractions and processing time for the entire load fractions. Finally, we also present a closed-form solution for scheduling polynomial loads with start-up delay in communication and computation. The numerical speedup results obtained using closed-form solution clearly show that super-linear speedup is possible for the polynomial computational loads.

Index Terms: Nonlinear divisible loads, broadcast communication or simultaneously load distribution model, overhead delays, single-level tree network.

1 Introduction

A divisible load is an input load that can be arbitrarily partitioned and assigned to distributed processors to gain the benefits of parallel and distributed processing. No precedence relationships between atomic loads of the entire payload are assumed in the divisible load theory. Atomic load cannot be divided further. Divisible loads are data parallel loads that are perfectly divisible amongst links and processors. Such loads arise in the parallel and distributed, and data intensive processing with massive amounts of data. Massively data parallel applications suitable for divisible load computing include grid computing, signal processing, image processing, multimedia computing, bio-intelligent computing, geological data processing, and aerospace data processing. Of course, arbitrarily divisible load applications relate to a narrow class of problems.

Cheng and Robertazzi [12] are the pioneers in the divisible load theory. Since 1988 hundreds of works by a number of researchers [1, 3–13, 15, 18, 19, 21–23, 25–27, 29–35] have developed algebraic means determining optimal fractions of a load distributed to processors via corresponding links under a given interconnection topology and a given scheduling policy. Here, optimality is defined in terms of speedup and execution time. Speedup is defined by the ratio of the execution time of the sequential algorithm to that of the parallel algorithm based on the divisible load theory. The theory to date largely involves loads of linear computational complexity [7, 8, 31]. In other words, computational or communication time is proportional to the size of fractional loads distributed to processors via corresponding links. Divisible load modeling should be of interest as it models, both computation and network communication in a completely integrated manner. A number of scheduling policies have been investigated including multi-installments [6] and multi-round scheduling [35], simultaneous distribution [19, 29], simultaneous start [25], detailed parameterizations and solution time optimization [1] and combinatorial schedule optimization [15]. Divisible loads may be divisible in fact or as an approximation as in the case of a large number of relatively small independent tasks [5, 9]. Many tutorials [7, 8, 31] on divisible load scheduling theory are available now.

Only a single level tree network (or star network) is considered in this paper as it forms a fundamental interconnection topology among many topologies such as linear daisy chains [12], buses [4,33], trees [3], hypercubes [10], and multidimensional meshes [11, 18]. 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. In short, the single level tree network is an archetypal network since it can generalize any topology [26].

There is an increasing amount of research on real-time modeling and simulation of complex systems such as image processing [14], etc. It is well known that many algorithms used in these areas require processing load of nonlinear complexity, i.e., the computational time of the given data/load is a nonlinear function of the load size (N). For example, line detection using Hough transform [14] and pattern recognition using 2D hidden Markov model (HMM) [28] requires $O(N^2)$ computational effort.The classical Hough transform was concerned with the identification of lines in the image, but later this transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The computational complexity for N points is approximately proportional to N^2. When, N is large, parallel or distributed processing is desired [17]. A separable 2D HMM for face recognition builds on an assumption of conditional independence in the relationship between adjacent blocks. This allows the state transition to be separated into vertical and horizontal state transitions. This separation of state transitions brings the complexity of the hidden layer of the proposed model from the order of $O(N^3k)$ to the order of $O(2N^2k)$, where N is the number of the states in the model and k is the total number of observation blocks in the image [17] and $O(2N^2k)$ in worst-case. In addition, we can also find real-world problems like molecular dynamic simulation of macromolecular systems, learning vector quantization neural network [24] and block tri-diagonalization of real symmetric matrices [2] which require second-order computational complexity.

In most of the algorithms which require nonlinear computational complexity, it is possible to divide the loads arbitrarily and process them independently such that the total processing time is less than the processing time on a single processor [13]. For some cases, we may need post-processing of the results obtained from the individual processors to get exactly the same solution as that of a solution obtained from a single processor. In this paper, we follow the fundamental assumption of arbitrarily divisible load [12] that the loads can be divided arbitrarily and can be processed without any precedence. Also, the post-processing time required to combined the results are assumed to be small and negligible. The computational complexity of the algorithm processing the computational load is nonlinear. Load is divided optimally by solving Equations (4) and (6).

Recently, Hung and Robertazzi [23] reiterate the nonlinear divisible load theory. Here, the computational loads require nonlinear processing time depending on the size of load fractions. In [23], the authors formulated the problem of scheduling such nonlinear computational loads in divisible load theory framework. We can determine the optimal load fractions assigned to the processors and the processing time by solving the nonlinear algebraic equations. But, solving these equations requires numerical methods and is computationally intensive to find optimal solution. Hence, we consider polynomial time complexity computational loads in a single-level tree network and present a closed-form solution for near optimal load fractions and processing time.

In this paper, we study the scheduling problem for computational loads of polynomial-order in a single level tree network with collective communication model (also known as simultaneous load distribution). Here, the polynomial time complexity computational load arrives at root processor and root processor distributes the load fractions simultaneously using collective broadcast model [16]. First, we formulate the scheduling problem for a general nth-order computational loads, i.e., processing time is $O(N^n)$. Using a mild assumption on the ratio of communication to computation time, we derive the closed-form expressions for optimal load fractions assigned to child processors and the processing time for second- and third-order cases (i.e., $\gamma = 2$ and $\gamma = 3$, respectively). The closed-form solution in this paper is an approximated solution because it is very difficult to derive an exact analytic solution. Due to the nonlinearity of the equations, it is not possible to derive the closed-form solution analytically. Only a numerical solution can be obtained. However, this paper provides an analytical closed-form solution by a simple approximation of the nonlinear equation. Numerical solutions are compared with the analytic solution to see if they conform to each other. The results clearly indicate that the analytical closed-form expression matches closely with the numerical solution. The numerical speedup results obtained using the closed-form solution clearly show that a super-linear speedup is possible for computational loads of polynomial-order. Super-linear speedup means a speedup of more than p when using p processors. Finally, we also present closed-form expressions for the load fractions and processing time in case of overhead factors in communication and computation links in addition to inherent computation and communication time.

First, nonlinear dependency in computational time function has been considered in [20]. The main contributions of this paper are,

- With a polynomial computation time function, we first derive closed-form expressions for load fractions assigned to the processors in the network and total load processing time. We also present the validity of the solution by comparing with analytic solution obtained using a numerical solver for Equations (4) and (6). The MATLAB nonlinear solver is used in this paper.
- With the proposed closed-form solutions, we can directly study the characteristics (processing time, speedup and ultimate performance bounds) of polynomial computational loads in a distributed system.
- Finally, we extend the results for the system with additive overheads in computation and communication in addition to inherent communication and computation time.

Section II describes the communication and computation model of divisible load theory considered in this paper. In addition, nomenclatures, definitions and notations are provided in this section. Section III derives the closed-form expressions for load fractions and processing time for a single level tree network with a broadcast communication model. This section also presents a numerical study to show super-linear speedup is possible when the computational time is nonlinear function of the size of load fractions. The study is extended to a system with overhead delays in communication and computation in Section IV. Section V concludes the paper.

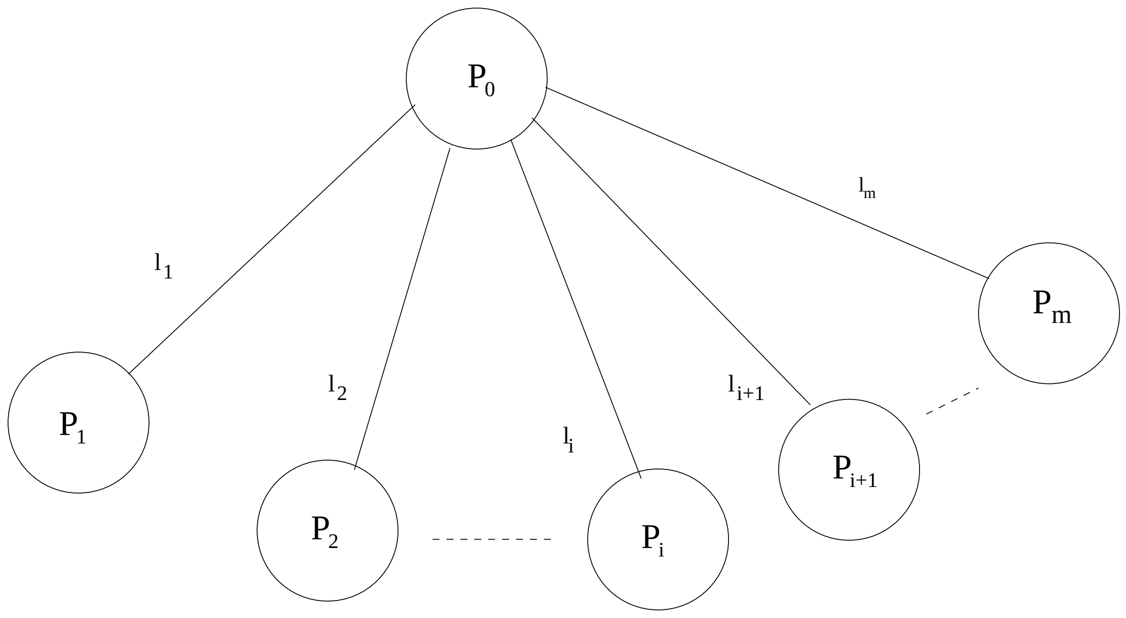


Figure 1: A single level tree network

2 Models and Notations

The single level tree network architecture considered in this paper is shown in Figure 1. Suppose there are m child processors $(p_1, p_2, \cdots, p_m)$ connected to a load originating processor (p_0) through communication links $(l_1, l_2, \cdots, l_m)$, as shown in Figure 1. The processor p_0 initially holds the processing load or computational load (N) and the processing load can be partitioned into $m + 1$ independent fractions

$(N\alpha_0, N\alpha_1, \cdots, N\alpha_m)$. Here, total size of processing load (N) is a very large quantity. The root processor p_0 keeps the load fraction $N\alpha_0$ for itself and distributes the remaining m fractions $(N\alpha_1, N\alpha_2, \cdots, N\alpha_m)$ to the child processors $(p_1, p_2, \cdots, p_m)$, concurrently. Note that the processor p_0 distributes the load fractions simultaneously, but not one by one (sequentially). Given that:

- The child processors start computing only after completely receiving the load fraction.
- The root processor p_0 can concurrently distribute the load fractions to its child processors.
- The load distribution process starts at time $t = 0$ and there is no time gap between communication and computation process.

Then, the finish time T_i for the processor p_i is the time difference between the time instant at which processor p_i starts receiving the load distribution and the time instant at which the processors stop computing.

$$T_i \quad = \quad T_i^{comm} + T_i^{comp}, \tag{1}$$

where T_i^{comm} is the load distribution time and T_i^{comp} is the computation time given as Equation (2). T_i is given as Equation (18), for example. These terms depend on the type of communication model and the nature of algorithm processing the computational load.

2.1 Collective Communication Model

In this paper, we use simultaneous broadcast ('All-Broadcast') model for concurrent transfer of load fractions to child processors. All-Broadcast is a collective communication model [16]. Using this collective communication model, the root processor p_0 can concurrently distribute/receive the load fractions to/from the child processors $p_1, p_2, \cdots, p_m$. For example, the processor p_0 can distribute the fractions $\{N\alpha_1, N\alpha_2, \cdots, N\alpha_m\}$ to the child processors $p_1, p_2, \cdots, p_m$ concurrently using the *broadcast* model [16]. In this model, the processor p_0 concurrently writes the load fractions $N\alpha_1$, $N\alpha_2 \cdots N\alpha_m$ into the respective communication buffer. The child processors read the data from the respective communication buffers.

The communication time required to completely receive the load fraction $N\alpha_i$ through the communication link l_i is the sum of constant start-up delay, θ_{cm}, and transmission time. The start-up time is a constant additive communication overhead component that includes the sum of all delays associated with the communication process. The transmission time is the time taken to read the data from the communication buffer by the processor p_i. This time depends on the communication link speed and transmission data size. Hence, the communication time, T_i^{comm}, can be written as follows:

$$T_i^{comm} = \theta_{cm} + (N\alpha_i)G_i, \tag{2}$$

where N is the size of the total processing load, and G_i is the time taken to read a unit data from the buffer.

2.2 Polynomial Computation Model

For many practical applications, the computation complexity of the algorithm processing the computational loads is nonlinear in problem size. Earlier studies in divisible load theory assume a linear computation complexity in their study. In this paper, we define the computational time function in terms of the running cost of an algorithm, the computing speed of unit load in the CPU and delay in extracting the actual load for processing.

The processor speed, p_i, in a network is modeled using the parameter A_i which is the time taken to process a unit load by the processor p_i. The constant start-up delay in processor p_i due to load extraction and processor initialization is modeled using the parameter θ_{cp}. The computation time T_i^{comp} required to process the load fraction $(N\alpha_i)$ is the sum of the constant additive start-up delay and nonlinear computation time as follows:

$$T_i^{comp} = \theta_{cp} + (N\alpha_i)^{\gamma} A_i, \tag{3}$$

where γ is an integer constant. The value of the integer constant (γ) depends on the nature of the algorithm used to process the load. The computation time function for any given processor is a polynomial equation in load fraction size. Equation (3) is completed using Equations (4) and (6).

2.3 Notation and Definitions

Notations:

N: The total size of the processing load.

m: Number of child processors.

γ: Integer constant depends on the nature of the algorithm used for processing the load. For example, this value is 2 for second-order nonlinear system (See Subsection 3.1).

α_i: Fraction of the processing load assigned to processor p_i, $i = 0, 1, \cdots, m$.

A_i: The computation speed parameter for processor p_i.

G_i: The communication speed parameter for link l_i.

θ_{cm}: A constant additive communication overhead component that includes the sum of all delays associated with the communication process.

θ_{cp}: A constant additive computation overhead component that includes the sum of all delays associated with the computation process.

Definitions:

T_i: **Finish time** for processor p_i is the time difference between the time instant at which the processor p_i stops computing and the time instant at which the root processor p_0 initiates the load distribution process.

T: **Processing time** is the time at which the entire load is processed; it is given by the maximum of the finish time of all processors; i.e., $T = max\{T_i\}$, $i = 0, 1, \cdots, m$, where T_i is the finish time of processor p_i.

3 Closed-Form Expression for Processing Time: Non-Affine Case

In this section, we present a closed-form expression for load fractions and processing time for non-affine case, i.e., the overhead factors are zero in communication and computational model. Now, we shall derive a closed-form expression for the processing time. For this purpose, we consider a heterogeneous single-level tree network in which the root processor p_0 uses a broadcast communication model to concurrently distribute the load fractions $(N\alpha_1, N\alpha_2, \cdots, N\alpha_m)$ to child processors $(p_1, p_2, \cdots, p_m)$. Here, we assume that the root processor p_0 can start its computation while its front-end distributes the load fractions concurrently. The process of load distribution is described using a timing diagram similar to a Gantt chart as shown in Figure 2. Without loss of generality, it is assumed that the root processor p_0 starts broadcasting the load fractions at time $t = 0$. In divisible load theory literature [7], it has been rigorously proved that for optimal processing time, all the processors involved in the computation of the processing load must stop computing at the same time instant. In this paper, we also use this optimality criterion.

From the timing diagram shown in Figure 2, the recursive equations for load distribution are

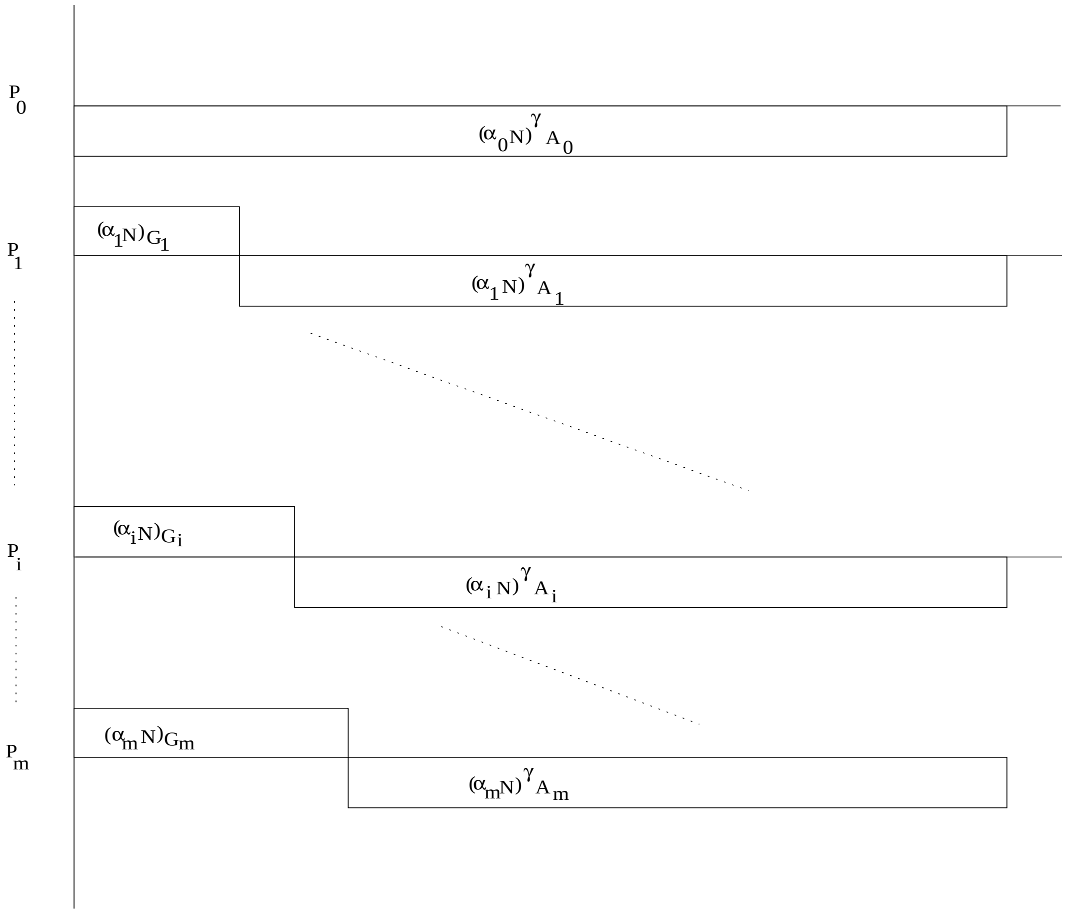
$$(\alpha_0 N)^\gamma A_0 \;=\; (\alpha_i N)^\gamma A_i + (\alpha_i N)\, G_i, \qquad i = 1, 2, \cdots, m. \tag{4}$$

Denoting $f_i = \frac{A_{i-1}}{A_i}$ and $\beta_i = \frac{G_i}{A_i}$, for all $i = 1, 2, \cdots, m$. Equation (4) can be rewritten as

$$(\alpha_i N)^\gamma + (\alpha_i N)\, \beta_i - (\alpha_0 N)^\gamma f_1 f_2 \cdots f_i \;=\; 0, \qquad i = 1, 2, \cdots, m, \tag{5}$$

Now, we see, from Equation (5), there are m nonlinear equations with $m+1$ variables, and together with the normalization equation, we have $m+1$ equations. The normalization equation is written as

$$1 \;=\; \sum_{i=0}^{m} \alpha_i.. \tag{6}$$



P0
$(\alpha_0 N)^\gamma A_0$
$(\alpha_1 N) G_1$
P1
$(\alpha_1 N)^\gamma A_1$
$(\alpha_i N) G_i$
Pi
$(\alpha_i N)^\gamma A_i$
$(\alpha_m N) G_m$
Pm
$(\alpha_m N)^\gamma A_m$

Figure 2: Timing diagram for load distribution process for non-affine case

In the earlier studies [12,25,26,31], divisible load theory has considered only the first-order case ($\gamma = 1$). In this case, these $m+1$ equations are solved by expressing each load fractions in Equation (5) in terms of α_0. Using normalization equation, we can obtain the closed-form expression for load fraction α_0. In a general case of the high-order functions (i.e., $\gamma \geq 2$), it is difficult to express each α_i in terms of α_0 and also difficult to find general solution. Hence, in the following subsection, first, we present the closed-form expressions for second-order ($\gamma = 2$) complexity and next we present the third-order complexity ($\gamma = 3$). Finally, we present approximate closed-form solutions for higher-order complexity.

3.1 Case A: Second-Order Complexity ($\gamma = 2$)

For $\gamma = 2$, the load fractions are expressed in quadratic equations. Applying the quadratic formula, we find the solution for the load fraction α_i in terms of α_0 from Equation (5) as follows:

$$\alpha_i \;=\; \frac{-\beta_i \pm \sqrt{\beta_i^2 + 4\left(\alpha_0 N\right)^2 \prod_{k=1}^{i} f_k}}{2N}, \quad i = 1, 2, \cdots, m. \tag{7}$$

Since, the load fractions α_i are positive and greater than or equal to zero, we consider only the positive real root as a solution as follows:

$$\alpha_i \;=\; \frac{-\beta_i + \sqrt{\beta_i^2 + 4\left(\alpha_0 N\right)^2 \prod_{k=1}^{i} f_k}}{2N}, \quad i = 1, 2, \cdots, m. \tag{8}$$

From the above we can see that there are m nonlinear equations with $m+1$ variables, and together with the normalization equation, we have $m + 1$ equations. Now, we will derive the closed-form expression for load fractions and processing time by expressing α_i $(i = 1, 2, \cdots, m)$ in terms of α_0 and normalization equation is used to solve for α_0.

By substituting Equation (8) in (6), we can get

$$2\left(\alpha_0 N\right) - \sum_{i=1}^{m} \beta_i + \sum_{i=1}^{m} \sqrt{\beta_i^2 + 4\left(\alpha_0 N\right)^2 \prod_{k=1}^{i} f_k} = 2N. \tag{9}$$

Note that Equation (9) is nonlinear and finding analytical solution is not possible. Hence, in this paper, we find an approximate solution without loosing accuracy. The square root term in Equation (9) is expressed using Taylor series as

$$\sqrt{\beta_i^2 + 4\left(\alpha_0 N\right)^2 \prod_{k=1}^{i} f_k} = 2(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k} + \frac{\beta_i^2}{4(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k}} + O\left(\left(\frac{\beta_i}{(\alpha_0 N)^2 \prod_{k=1}^{i} f_k}\right)^2 .\right)$$

The higher-order terms are function of $\frac{\beta_i^2}{4(\alpha_0 N)^2 \prod_{k=1}^{i} f_k}$. The term in denominator is much higher than the communication to computation speed ratio (β_i), i.e., $\alpha_0 N >>$

1, $f_i > 1$ and $\beta_i < 1$. Hence, the higher-order terms can be neglected without sacrificing much of accuracy in the solution. Hence, the square root term can be approximated as

$$\sqrt{\beta_i^2 + 4\left(\alpha_0 N\right)^2 \prod_{k=1}^{i} f_k} = 2(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k} + \frac{\beta_i^2}{4(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k}}. \tag{10}$$

By substituting Equation (10) in Equation (9), we get

$$2(\alpha_0 N) - \sum_{i=1}^{m} \beta_i + \sum_{i=1}^{m} \left(2(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k} + \frac{\beta_i^2}{4(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k}} \right) = 2N. \tag{11}$$

Equation (11) is reduced to

$$8A(m)\left(\alpha_0 N\right)^2 - 4B(m)\left(\alpha_0 N\right) + C(m) \;=\; 0, \tag{12}$$

where $f_0 = 1$, and

$$\begin{aligned} A(m) &= \left(\prod_{k=1}^{m} \sqrt{f_k} \right) \cdot \left(1 + \sum_{i=1}^{m} \prod_{i}^{k=1} \sqrt{f_k} \right), & (13) \\ B(m) &= \left(2N + \sum_{j=1}^{m} \beta_j \right) \prod_{k=1}^{m} \sqrt{f_k}, & (14) \\ C(m) &= \prod_{k=1}^{m} \sqrt{f_k} \cdot \left(\sum_{i=1}^{m} \frac{\beta_i^2}{\prod_{k=1}^{i} \sqrt{f_k}} \right). & (15) \end{aligned}$$

Since Equation (12) is quadratic in terms of $(\alpha_0 N)$, we can write the solution for α_0 as follows:

$$\alpha_0 \;=\; \frac{B(m) \pm \sqrt{B(m)^2 - 2A(m)C(m)}}{4NA(m)}. \tag{16}$$

From Equations (14) and (15), we can show that $B(m) > 0$ and $C(m) > 0$. In other words, $B(m)C(m) > 0$. In addition, from Equations (13), (14), and (15), we can show that $B^2(m) \geq A(m)C(m)$. Even the term $A(m)$ is greater than the term $B(m)$, most of the elements in the term $C(m)$ are the sum of β_i^2. Since $\beta_i \ll 1$,

Table 1: Parameters of Single Level Tree Network: Non-Affine Case

Parameter	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
A_i	6.3	6.6	6.9	7.2	7.5	7.8	8.1	8.4	8.7	9.0
G_i	-	0.60	0.70	0.71	0.72	0.73	0.8	0.81	0.82	0.90

it is obvious that $B(m)^2 > 2A(m)C(m)$. Hence, the solution for load fraction α_0 assigned to the root processor p_0 is given as follows:

$$\alpha_0 = \frac{B(m) + \sqrt{B(m)^2 - 2A(m)C(m)}}{4NA(m)}. \tag{17}$$

Total load processing time $T(m)$ is given as

$$T(m) = \left(\frac{B(m) + \sqrt{B(m)^2 - 2A(m)C(m)}}{4A(m)}\right)^2 A_0. \tag{18}$$

Using the closed-form expressions for load fractions and total load processing time, one can easily determine the speed for any given load size and number of processors. Now, we present an numerical example to illustrate the closeness of the solution obtained using the closed-form expression and solution obtained using numerical methods.

Numerical Example I: Consider a single level tree network with 10 processors and 9 links. The parameters of the processors and links are given in Table 1. The total load (N) assumed to be processed is 10.

By using our closed-form expression in Equation (18) for processing time, we can obtain the total load processing time directly for different numbers of processors. Here, we also used the numerical solver in MATLAB to obtain the actual algebraic solution. The processing time for various number of processors obtained using the proposed closed-form expression and actual algebraic solution are given in Table 2. From the table, we can see that the direct solution obtained using the proposed

Table 2: Total Load Processing Time for Various Number of Processors for a Second-Order System

m	Processing Time		Absolute
	Proposed Solution	Numerical Solution	Difference
1	162.646	162.644	2.159e-03
2	74.663	74.664	1.146e-03
3	43.332	43.332	4.724e-05
4	28.584	28.582	1.754e-03
5	20.441	20.441	5.039e-04
6	15.462	15.460	2.000e-03
7	12.178	12.177	9.015e-04
8	9.891	9.889	1.555e-03
9	8.235	8.234	1.667e-03

closed-form expression is matching with the actual solution and the error is only in the third decimal. Hence, we can say that neglecting higher-order terms does not influence the accuracy significantly in the closed-form expression.

3.1.1 Homogeneous System

As a special case, we present a homogeneous system, where $A_i = A$ and $G_i = G$ for all i. Hence, the value of f_i is 1 and all β_i is identical such that $\beta_i = \beta$. Then, from Equation (8), we can obtain the load fractions assigned to child processors for the homogeneous case as follows:

$$\alpha_i = \frac{-\beta + \sqrt{\beta^2 + 4\left(\alpha_0 N\right)^2}}{2N}, \quad i = 1, 2, \cdots, m. \tag{19}$$

Using the normalization equation ($\sum_{i=0}^{m} \alpha_i = 1$) in Equation (6), we can obtain the closed-form expression for the load fraction α_0 as follows:

$$\alpha_0 + \sum_{i=1}^{m} \frac{-\beta + \sqrt{\beta^2 + 4\left(\alpha_0 N\right)^2}}{2N} = 1. \tag{20}$$

Equation (20) reduces by using the Taylor expansion of the square-root term as

$$(m^2-1)(\alpha_0 N)^2 + (2N + m\beta)(\alpha_0 N) - N(N + m\beta) = 0. \tag{21}$$

The positive root of the above equation is the load fraction assigned to root processor p_0, which is given as

$$\alpha_0 = \frac{-(2N + m\beta) + m\sqrt{\beta^2 + 4N(N + m\beta)}}{2N\,(m^2-1)}. \tag{22}$$

The total load processing time is

$$T(m) \;=\; \left(\frac{-(2N + m\beta) + m\sqrt{\beta^2 + 4N(N + m\beta)}}{2\,(m^2-1)}\right)^2 A. \tag{23}$$

When the communication time is smaller than the computation time, asymptotic solution for $T(m)$ is obtained as

$$\lim_{\beta\to 0} T(m) \;=\; \frac{N^2}{(m+1)^2}A. \tag{24}$$

Two Processor System: Equation (22) holds if $m > 1$. Thus, Equation (22) cannot be used for a two-processor case with $m = 1$. For a two-processor system, the load fractions for each processor are obtained by substituting $\alpha_1 = 1 - \alpha_0$ from Equation (6) with Equation (5) as follows:

$$\alpha_0 \;=\; \frac{N+\beta}{2N+\beta}, \tag{25}$$

and

$$\alpha_1 \;=\; \frac{N}{2N+\beta}. \tag{26}$$

Total load processing time is

$$T(1) \;=\; \left(\frac{N+\beta}{2N+\beta}N\right)^2 A. \tag{27}$$

Asymptotic Analysis: We use the above closed-form solution to obtain the ultimate performance limits of the network with respect to the number of processors. In case of polynomial computing loads, the total computing load (N) can be divided into N fractions and processed individually in N processors. Hence, the maximum number of processors used in the network is $m = N - 1$.

The total load processing time is

$$T(N-1) = \left(\frac{-(2N + (N-1)\beta) + (N-1)\sqrt{\beta^2 + 4N(N + (N-1)\beta)}}{2\left((N-1)^2 - 1\right)} \right)^2 A. \quad (28)$$

The speedup is given as

$$\begin{aligned} S(N-1) &= \frac{N^2 A}{T(N-1)} \\ &= \frac{4N^2((N-1)^2-1)^2}{\left(-(2N + (N-1)\beta) + (N-1)\sqrt{\beta^2 + 4N(N + (N-1)\beta)}\right)^2}. \end{aligned} \quad (29)$$

When the communication time is very small ($\beta \rightarrow 0$), then the ultimate speedup is

$$\lim_{\beta \rightarrow 0} S(N-1) = N^2. \quad (30)$$

From Equation (30), we can achieve a super-linear speedup when the processing time is a nonlinear function of the size of load fractions. In Figure 3, we plot the speedup against number of processors for different values of communication to computation ratio (β). From the figure, we can see that the performance curve is bounded between 0 for $\beta \rightarrow \infty$ and N^2 when $\beta \rightarrow 0$. The figure clearly indicates that the speedup characteristics pose a super-linear behavior. Since the speedup figures ares much larger than the number of processors, we can say that super-linear performance is achieved.

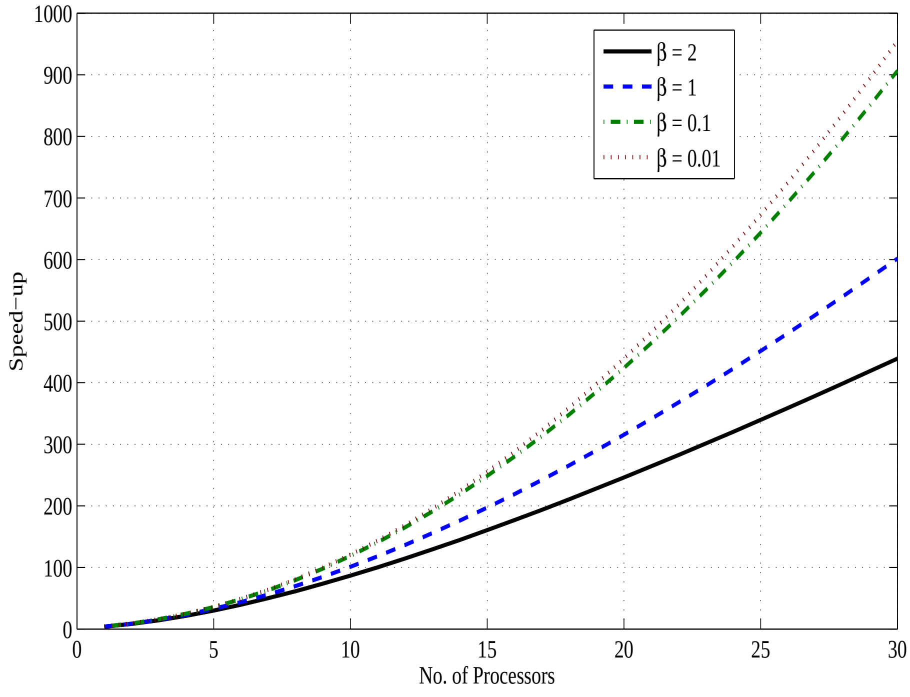


Figure 3: Speedup vs. number of processors

3.2 Case B: Third-Order Complexity ($\gamma = 3$)

For $\gamma = 3$, the load fractions are expressed in cubic equations. From the timing diagram or from Equation (5), the load distribution equations for third-order complexity is given below.

$$(\alpha_i N)^3 + (\alpha_i N)\,\beta_i = (\alpha_0 N)^3 \prod_{k=1}^{i} f_k, \quad i = 1, 2, \cdots, m. \tag{31}$$

Using the cubic equation, we can express the load fraction α_i in terms of load fraction α_0. For the cubic equation of this form $x^3 + ax - by^3 = 0$, we get one real and two complex roots. Since the load fraction assigned to the processors in the network is real and positive number, we only consider the real root.

$$(\alpha_i N) = \frac{\left(9by^3 + \sqrt{3}\sqrt{4a^3 + 27b^2y^6}\right)^{1/3}}{2^{1/3}3^{2/3}} - \frac{(2/3)^{1/3}\, a}{\left(9by^3 + \sqrt{3}\sqrt{4a^3 + 27b^2y^6}\right)^{1/3}}, \tag{32}$$

where $a = \beta_i$, $b = \prod_{k=1}^{i} f_k$ and $y = \alpha_0 N$.

First, we consider the term inside the cubic root for determining the closed-form expression. The cubic root term can be expanded using the Taylor series as

$$\left(9by^3 + \sqrt{3}\sqrt{4a^3 + 27b^2y^6}\right)^{1/3} = \left[9by^3 + 9by^3\left[1 + \frac{2a^3}{27b^2y^6} + O\left(\frac{a^6}{b^4y^{12}}\right)\right]\right]^{1/3}. \tag{33}$$

Here, the higher-order terms are a function of 6^{th} power of communication to computation speed ratio, 4^{th} power of processing speed ratio, and 12^{th} power of $\alpha_0 N$. In general, the communication time is less than computation time ($\beta_i < 1$). Otherwise, the processor will not participate in the load distribution process [7]. Also, the load fraction assigned to the root processor ($\alpha_0 N$) and processor speed ratio is greater than one. Hence, we can neglect the higher-order terms from the above equation.

By neglecting the higher-order term in Equation (33) and substitute in Equation (32), we get

$$(\alpha_i N) = \prod_{k=1}^{i} (f_k)^{1/3} (\alpha_0 N) - \frac{\beta_i}{3\prod_{k=1}^{i} (f_k)^{1/3} (\alpha_0 N)}. \tag{34}$$

Substituting Equation (34) in Equation (6), we get

$$(\alpha_0 N)^2 A(m) - N(\alpha_0 N) - C(m) = 0, \tag{35}$$

where

$$\begin{aligned} A(m) &= 1 + \sum_{i=1}^{m} \prod_{k=1}^{i} (f_k)^{1/3}, \\ C(m) &= \sum_{i=1}^{m} \frac{\beta_i}{3\prod_{k=1}^{i} (f_k)^{1/3}}. \end{aligned}$$

The above quadratic equation can be easily solved to determine the closed-form expression for load fraction α_0 and is

$$\alpha_0 = \frac{N + \sqrt{N^2 + 4A(m)C(m)}}{2NA(m)}. \tag{36}$$

The total load processing time for third-order complexity is

$$T(m) = \left[\frac{N + \sqrt{N^2 + 4A(m)C(m)}}{2A(m)}\right]^3 A_0. \tag{37}$$

Using the closed-form expression, we can directly determine the processing time and analyze the characteristics of the third-order system.

3.2.1 Asymptotic Analysis

We use the above closed-form expression to obtain the ultimate performance limits for the third-order complexity with respect to the number of processors. For this, we consider the homogeneous system, i.e., $A_i = A$ and $G_i = G$ for all $i = 1, 2, \cdots, m$. In this case, the closed-form solution for processing time given in Equation (37) can be written as

$$T(m) = \left[\frac{N + \sqrt{N^2 + \frac{4}{3}m(m+1)\beta}}{2(m+1)}\right]^3 A. \tag{38}$$

Hence, the maximum number of processors used in the network is $m = N - 1$ and the processing time is

$$T(N-1) = \left[\frac{N + \sqrt{N^2 + \frac{4}{3}N(N-1)\beta}}{2N}\right]^3 A. \tag{39}$$

Ultimate speedup performance for the third-order complexity is

$$\begin{aligned} S(N-1) &= \frac{N^3 A}{T(N-1)} \\ &= \frac{8N^6}{\left[N + \sqrt{N^2 + \frac{4}{3}N(N-1)\beta}\right]^3}. \end{aligned} \tag{40}$$

When the communication to computation ratio tends to be a small value then we achieve the upper bound on speedup as follows:

$$\lim_{\beta \to 0} S(N-1) \;=\; N^3. \tag{41}$$

From Equation (41), we can achieve a super-linear speedup when the processing time is a nonlinear function of the size of load fractions.

3.3 Case C: Higher-Order Complexity

For general higher-order system it is difficult to derive a closed-form expression. Hence, in this section, we attempt to solve this problem for a specific situation. From the timing diagram or from Equations (4) and (6), we can rewrite the load distribution equation as

$$(\alpha_0 N)^\gamma A_0 \;=\; (\alpha_i N)^\gamma A_i + (\alpha_i N)\, G_i, \quad i = 1, 2, \cdots, m. \tag{42}$$

Divide the above equation by $N^\gamma A_i$, we get

$$(\alpha_0)^\gamma \prod_{k=1}^{i} f_k \;=\; (\alpha_i)^\gamma + (\alpha_i)\, \eta_i, \quad i = 1, 2, \cdots, m. \tag{43}$$

where $\eta_i = \frac{\beta_i}{N^{\gamma-1}}$.

Without loss of generality, we can assume that the communication link speed is less than the computation speed (at least of the order of 10, $\beta_i < 0.1$). In addition, the total size of polynomial computing load is large and also the order complexity (γ) is greater than 4. Hence, the term η_i becomes infinitesimal and can be neglected. Under this condition, the Equation (43) is reduced to

$$(\alpha_0)^\gamma \prod_{k=1}^{i} f_k \;=\; (\alpha_i)^\gamma, \quad i = 1, 2, \cdots, m. \tag{44}$$

The load fraction α_i can be expressed in terms of α_0 as

$$\alpha_i = \alpha_0 \prod_{k=1}^{i} f_k^{1/\gamma}, \quad i = 1, 2, \cdots, m. \tag{45}$$

Using the normalization process using Equation (5), we can obtain the closed-form expression for α_0 as

$$\alpha_0 = \frac{1}{1 + \sum_{i=1}^{m} \prod_{k=1}^{i} f_k^{1/\gamma}}. \tag{46}$$

From the timing diagram shown in Fig. 2, the total load processing time is

$$T(m) = (\alpha_0 N)^\gamma A_0 = \left[\frac{N}{1 + \sum_{i=1}^{m} \prod_{k=1}^{i} f_k^{1/\gamma}}\right]^\gamma A_0. \tag{47}$$

Using this closed-form expression, we can study the behavior of the general polynomial computation loads. For homogeneous system ($A_i = A$ and $G_i = G$), the total load processing time is reduced to

$$T(m) = \left[\frac{N}{1+m}\right]^\gamma A. \tag{48}$$

Hence, the ultimate bound on speedup will be m^γ, which is similar to the second and third-order case.

4 Closed-Form Expression for Processing Time: Affine Case

In this section, we consider the presence of additive overhead component in the communication and computation model. Similar to the case without overhead (non-affine model), here also, the load distribution process is described using the Gantt chart like timing diagram as shown in Figure 4. Also, assume that all processors stop computing at the same time to achieve the optimal solution.

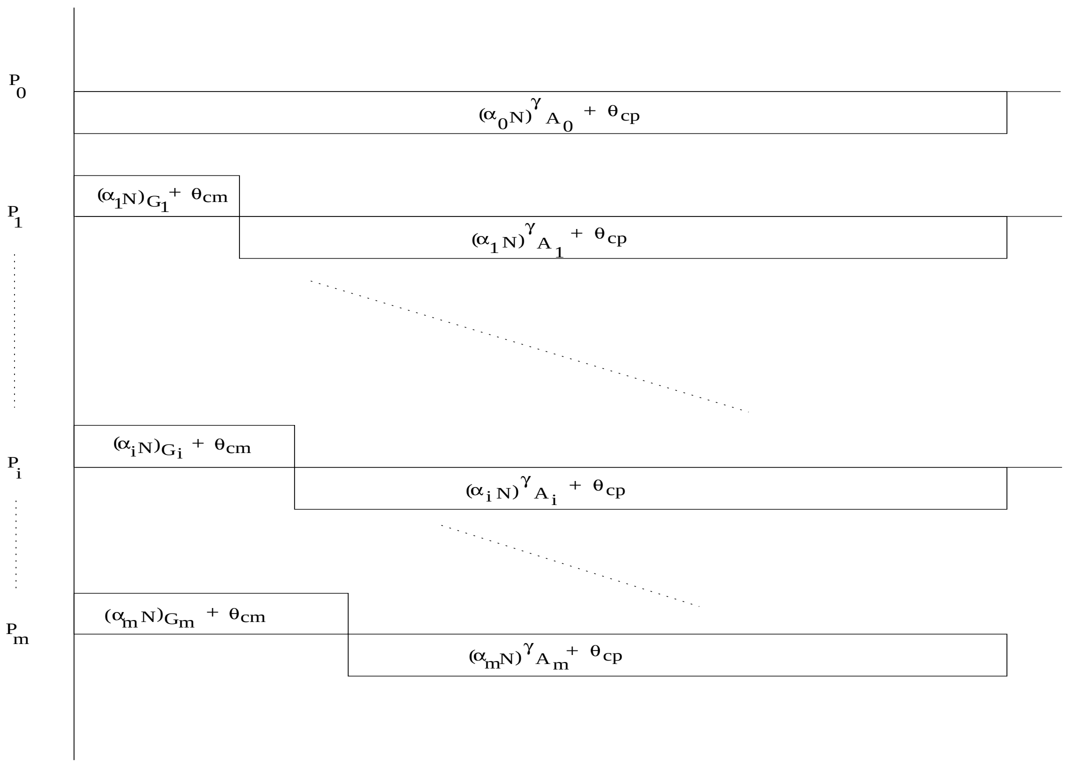

P_0
$(\alpha_0 N)^\gamma A_0 + \theta_{cp}$
$(\alpha_1 N) G_1 + \theta_{cm}$
P_1
$(\alpha_1 N)^\gamma A_1 + \theta_{cp}$
$(\alpha_i N) G_i + \theta_{cm}$
P_i
$(\alpha_i N)^\gamma A_i + \theta_{cp}$
$(\alpha_m N) G_m + \theta_{cm}$
P_m
$(\alpha_m N)^\gamma A_m + \theta_{cp}$

Figure 4: Timing diagram for polynomial load distribution: Affine case

From the timing diagram, we can write the load distribution equations as follows,

$$(\alpha_0 N)^\gamma A_0 + \theta_{cp} \quad = \quad (\alpha_i N)^\gamma A_1 + \theta_{cp} + (\alpha_i N) G_i + \theta_{cm}, \quad i = 1, 2, \cdots, m, \tag{49}$$

where θ_{cp} and θ_{cm} are the constant additive start-up delays in computation and communication respectively.

Denoting $f_i = \frac{A_{i-1}}{A_i}$, $\beta_i = \frac{G_i}{A_i}$ and $\delta_i = \frac{\theta_{cm}}{A_i}$, the above equations reduce to

$$(\alpha_i N)^\gamma + (\alpha_i N) \beta_i + \delta_i \quad = \quad (\alpha_0 N)^\gamma \prod_{k=1}^{i} f_k, \quad i = 1, 2, \cdots, m \tag{50}$$

and the normalization equation is

$$\sum_{i=0}^{m} \alpha_i \quad = \quad 1. \tag{51}$$

First, we present the closed-form expression derivations for second-order complexity ($\gamma = 2$). Finding closed-form expression for higher-order is difficult for affine case.

4.1 Case A: Second Order Complexity

By substituting $\gamma = 2$ in Equation (50), we get a following second-order equation.

$$(\alpha_i N)^2 + (\alpha_i N)\beta_i + \delta_i = (\alpha_0 N)^2 \prod_{k=1}^{i} f_k, \quad i = 1, 2, \cdots, m. \tag{52}$$

By applying the quadratic formula, we can express the load fraction (α_i) in terms of α_0 as follows:

$$(\alpha_i) = \frac{-\beta_i + \sqrt{\beta_i^2 + 4(\alpha_0 N)^2 - 4\delta_i}}{2N}, \quad i = 1, 2, \cdots, m. \tag{53}$$

Since, the load fractions (α_i) are a real positive root, we consider only the positive real root.

By substituting the load fractions (α_i) in normalization equation, we get

$$2(\alpha_0 N) - \sum_{i=1}^{m} \beta_i + \sum_{i=1}^{m} \left(\beta_i^2 - 4\delta_i + 4(\alpha_0 N)^2 \prod_{k=1}^{i} f_k \right)^{1/2} = 2N. \tag{54}$$

Using first-order approximation for the square root term as explained in non-affine case, we can rewrite the above equation as

$$2(\alpha_0 N) - \sum_{i=1}^{m} \beta_i + \sum_{i=1}^{m} \left(2(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k} + \frac{\beta_i^2 - 4\delta_i}{4(\alpha_0 N) \prod_{k=1}^{i} \sqrt{f_k}} \right) = 2N. \tag{55}$$

The above equation is reduced to

$$8A(m)(\alpha_0 N)^2 - 4B(m)(\alpha_0 N) + C(m) = 0, \tag{56}$$

where

$$A(m) = \left(\prod_{k=1}^{m} \sqrt{f_k} \right) \cdot \left(1 + \sum_{i=1}^{m} \prod_{i}^{k=1} \sqrt{f_k}, \right) \tag{57}$$

$$B(m) = \left(2N + \sum_{j=1}^{m} \beta_j\right) \prod_{k=1}^{m} \sqrt{f_k}, \tag{58}$$

$$C(m) = \prod_{k=1}^{m} \sqrt{f_k}. \left(\sum_{i=1}^{m} \frac{\beta_i^2 - 4\delta_i}{\prod_{k=1}^{i} \sqrt{f_k}}.\right) \tag{59}$$

From the solution of the quadratic equation, we can derive the closed-form expression for load fraction α_0 as

$$\alpha_0 = \frac{B(m) + \sqrt{B(m)^2 - 2A(m)C(m)}}{4NA(m)}. \tag{60}$$

The total load processing time $T(m)$ is

$$T(m) = \left[\frac{B(m) + \sqrt{B(m)^2 - 2A(m)C(m)}}{4A(m)}\right]^2 A_0. \tag{61}$$

Using the closed-form expressions for load fractions and processing time, similar to non-affine case, we can also analyze the characteristics of second-order computational complexity. By comparing the expression for total load processing time in affine (Eq. 61) and non-affine case (Eq. 18), we can see that one can obtain the closed-form expression for total load processing time in affine case by substituting the term β_i^2 in $C(m)$ by $\beta_i^2 - 4\delta_i$.

4.1.1 Numerical Example II

For the sake of simplicity, we consider the same parameters used in the numerical example I. The network has 10 processors and 9 links. The total load (N) assumed to be processed is 10. Assume the value 1 for both θ_{cp} and θ_{cm}.

By using our closed-form expression in Equation (61) for processing time, we can obtain the solution for different numbers of processors. The actual numerical solution for processing time and processing time obtained using the closed-form expression

Table 3: Total Load Processing Time for Various Number of Processors: Affine Case

m	Processing Time		Absolute
	Proposed Solution	Numerical Solution	Difference
1	228.774	228.744	2.047e-04
2	103.661	103.661	8.443e-04
3	59.681	59.679	2.327e-03
4	39.232	39.233	7.976e-04
5	28.074	28.073	7.910e-04
6	21.315	21.315	2.847e-04
7	16.908	16.908	2.150e-04
8	13.872	13.872	4.124e-04
9	11.690	11.689	8.983e-04

are given in Table 3. From the table, we can clearly see that the processing time obtained using the proposed closed-form expression closely matches the actual solution.

5 Conclusions

The problem of distributing a nonlinear divisible load on a single level tree network with collective communication model is presented in this paper. Here, a polynomial computation load with computational complexity proportional to its processing load size is considered. A closed-form solution for the load fractions assigned to the processors and the processing time are derived from the nonlinear recursive equations. Numerical solutions are compared with the analytic solutions for its closeness. The results clearly show that the proposed closed-form expression matches with the analytic solutions. Using the closed-form expression, we can directly analyze the speedup behavior and ultimate bounds on the speedup. The results clearly indicate that the speedup achieves a super-linear behavior for the polynomial computation loads. The study is also extended to the case with additive overhead components in communication and computational time.

Acknowledgments

The first author would like to acknowledge the support by the NTU-SUG program by Nanyang Technological University. This work was in part supported by 3DLife, CTRC, and ITRC from Korea.

References

[1] Adler, M., Gong, Y., and Rosenberg, A. L., "Optimal sharing of bags of tasks in heterogeneous clusters," *Proceedings of the Annual ACM Symposium on Parallel algorithms and Architectures,* San Diego, California, USA, pp. 1-10, 2003.

[2] Bai, Y. and Robert, R. C., "Parallel block tridiagonalization of real symmetric matrices," *Journal of Parallel and Distributed Computing,* vol. 68, pp. 703-715, 2008.

[3] Barlas, G. D., "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, 1998.

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

[5] Beaumont, O., Carter, L., Ferrante, J., Legrand, A., and Robert, Y., "Bandwidth-centric allocation of independent tasks on heterogeneous plat-

forms," *Proceedings of the International Parallel and Distributed Processing Symposium*, IEEE Computer Society Press, 2002.

[6] Bharadwaj, V., Ghose, D., and Mani, V., "Multi-installment load distribution in tree networks with delay," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 31, pp. 555-567, 1995.

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

[8] Bharadwaj, V., Ghose, D., and Robertazzi, T. G., "Divisible load theory: A new paradigm for load scheduling in distributed systems," *Cluster Computing,* vol. 6, pp. 7-18, 2003.

[9] Bharadwaj, V., Viswanadham, N., "Suboptimal solutions using integer approximation techniques for scheduling divisible loads on distributed bus networks," *IEEE Transactions on System, Man, and Cybernetics-Part A: Systems and Humans,* vol. 30, pp. 680-691, 2000.

[10] Blazewicz, J., and Drozdowski, M., "Scheduling divisible jobs on hypercubes," *Parallel Computing,* vol. 21, pp. 1945-1956, 1995.

[11] Blazewicz, J., Drozdowski, M., Guinand, F., and Trystram, D., "Scheduling a divisible task in a 2-dimensional mesh," *Discrete Applied Mathematics*, vol. 94, pp. 35-50, 1999.

[12] Cheng, Y. C., and Robertazzi, T. G., "Distributed computation with communication delays," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 24, pp. 700-712, 1988.

[13] Drozdowski, M., and Wolniewicz, P., ”Out-of-core divisible load processing,” *IEEE Tranactions on Parallel and Distributed Systems,* vol. 14, pp. 1048-1056, 2003.

[14] Duda, R. O. and Hart, P. E., ”Use of the Hough transformation to detect lines and curves in pictures,” *Communications of the ACM*, vol. 15, pp. 11-15, 1972.

[15] Dutot, P.-F., ”Divisible load on heterogeneous linear array,” *Proceedings of the International Parallel and Distributed Processing Symposium*, Nice, France 2003.

[16] W. Gropp, W., Lusk, E., and Skjellum, A., *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, 1994.

[17] Guil, N., Villalba, J., and Zapata, E. L., ”A fast Hough transform for segment detection,” *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1541-1548, 1995.

[18] Glazek, W., ”A multistage load distribution strategy for three dimensional meshes,” *Cluster Computing,* vol. 6, pp. 31-40, 2003.

[19] Hung, J. T., Kim, H. J., and Robertazzi, T. G., ”Scalable scheduling in parallel processors,” *Proceedings of the Conference on Information Sciences and Systems*, Princeton University, Princeton, NJ, 2002.

[20] Hung, J. T. and Robertazzi, T., ”Distributed scheduling of nonlinear computational loads,” *Proceedings of the Conference on Information Sciences and Systems*, The Johns Hopkins University, March 12-14, 2003.

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

[22] Hung, J. T., and Robertazzi, T. G., "Scalable scheduling for clusters and grids using cut through switching," *International Journal of Computers and Their Applications,* vol. 26, pp. 147-156, 2004.

[23] Hung, J. T., and Robertazzi, T. G., "Scheduling nonlinear computational loads," *IEEE Trans. On Aerospace Electronics and Systems*, vol. 44, no. 3, pp. 1169-1182, 2008.

[24] Khalifa, K. B., Boubaker, M., Chelbi, N., and Bedoui, M. H., "Learning vector quantization neural network implementation using parallel and serial arithmetic," *International Journal of Computer Sciences and Engineering Systems*, vol. 2, No. 4, pp. 251-256, 2008.

[25] Kim, H. J., "A novel load distribution algorithm for divisible loads," *Cluster Computing,* vol. 6, pp. 41-46, 2003.

[26] Kim, H. J., Jee, G.-I., and Lee, J. G., "Optimal load distribution for tree network processors," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 32, pp. 607-612, 1996.

[27] Li, K., "Parallel processing of divisible loads on partitionable static interconnection networks," *Cluster Computing*, vol. 6, pp. 47-56, 2003.

[28] Othman, H. and Aboulnasr, T., "A separable low complexity 2D HMM with application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1229-1238, 2003.

[29] Piriyakumar, D. A. L., and Murthy, C. S. R., "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, 1998.

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

[31] Robertazzi, T. G., "Ten reasons to use divisible load theory," *Computer*, vol. 31, pp. 63-68, 2003.

[32] Suresh, S., Mani, V., Omkar, S. N., and Kim, H. J., "Divisible load scheduling in distributed system with buffer constraints: Genetic algorithm and linear programming approach," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 21, no. 5, pp. 303-321, 2006.

[33] Suresh, S., Omkar, S. N., and Mani, V., "The effect of start-up delays in scheduling divisible loads on bus networks: An alternate approach", *Computer and Mathematics with Applications*, vol. 46, no. 10-11, pp. 1545-1557, 2003.

[34] Suresh, S., Omkar, S. N., and Mani, V., "Parallel implementation of back-propagation algorithm in networks of workstations", *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 1, pp. 24-34, 2005.

[35] Yang, Y., and Casanova, H., "UMR: A multi-round algorithm for scheduling divisible workloads," *Proceedings of the International Parallel and Distributed Processing Symposium*, Nice, France, 2003.