UNIVERSITY AT STONY BROOK

CEAS Technical Report 765

Record Search Time Evaluation using Divisible Load Analysis

Kwangil Ko and Thomas G. Robertazzi

Sept. 8, 1998

Record Search Time Evaluation using Divisible Load Analysis

Kwangil Ko and Thomas G. Robertazzi, Senior Member IEEE

Dept. of Electrical and Computer Engineering

University at Stony Brook

Stony Brook, NY 11794

Phone: 516-632-8412/8400

Fax: 516-632-8494

E-mail: tom@sbee.sunysb.edu

September 3, 1998

ABSTRACT

Divisible scheduling theory is used to solve for the expected time for searching for both single and multiple data records in parallel database architecture. The target architectures examined for illustrative purposes are the linear daisy chain and single level tree network with single and multiple installment load distribution. The use of divisible load modeling and analysis yields elegant expressions for expected search time

Index Terms

Daisy Chain
Database
Divisible Load Analysis
Performance Evaluation
Record Search Time
Tree Network

I. INTRODUCTION

Much of the computer revolution that is invisible to the public involves our increasing ability to create, manipulate and search very large amounts of information. Computer systems which do this are known as databases. The amount of data available to individuals or organizations is growing rapidly each year. Databases can be implemented on any platform including PC's, workstations, minicomputers and mainframes. A variety of software specially designed for database systems is also available. However, it has long been known that conventional computer systems can not satisfy the computational power requirement of many recent applications. It has been estimated that there are 20,000 U.S. government databases. Even twelve years ago, the Patent Office has a database of 25 terabytes of data [1]. Conventional digital computers using typical software approaches do not satisfy the requirements for efficient text retrieval. A software implementation of direct search on IBM 370/158 can process approximately 100,000 characters per second. But even if this speed could be increased tenfold, it would take 7,000 hours to search the 25 terabytes in the U.S. Patent Office's data file system. To decrease the searching time, the platform should ideally be a high performance platform such as a multi-processor or parallel computer having a collection of processors that cooperate and communicate to solve large problems quickly.

The main advantage of parallel processing is improved speed. This speed depends on the performance of parallel computation, communication speed, the ways in which a load is distributed to each processor and so on. The paradigm of divisible loads has been considered during the past decade as a tool for understanding parallel system performance. Divisible loads are loads which can be arbitrary partitioned among a number of processors. Scheduling

divisible load in parallel system is studied in [4]. The computation of schedules providing the minimal amount of solution time for linear daisy chain networks of communicating processors is examined in [6] [8] [9]. An example of the inclusion of solution time, the time taken for processors to report the solution back to the originator, is also presented. The determination of the optimal division of processing load is discussed for both tree networks with front end processors and tree networks without front end processors in [2] [5] [7]. Optimal load allocation for load sharing a divisible job over processors interconnected by bus networks is considered in [2] [3] [10].

The use of divisible load modeling and the associated analysis to evaluate parallel systems has a number of advantages. Linear and continuous modeling results in a tractable analysis. The underlying load allocation equations are deterministic so no probabilistic assumptions are made in this part of the analysis. Furthermore the model is generic enough that it can accommodate changes in technology and topology.

In this paper elegant expressions for the expected time to find both single and multiple records are found. The architectures investigated are a linear daisy chain of processor and a single level tree network. For the single level tree network both single installment and multi-installment load distribution are considered. These target architectures are chosen to be illustrative. The techniques described here can be used to model and solve for record search time on other architectures. This work is significant for demonstrating the power of divisible load scheduling theory for predicting search times.

II. The System Model

A parallel machine consists of a number of processors and an interconnection network to tie them together. This work examines a specific parallel processing problem on specific architectures that allows the study of the integration of communication and computation.

The situation to be considered involves a linear daisy chain of processors and a single level tree network, as are illustrated in Figure 1 and 2 respectively. The model of a linear daisy chain network or a single level tree network consist of (M+1) processors. Processor 0 is assumed to be the originating processor which distribute the fractions of the entire load to M processors. Without loss of generality, it will be assumed that the load is instantaneously delivered to the processor 0 and this processor becomes the load origination processor. Each processor is interfaced with the network via a front-end communication processor for communication off-loading. That is, the processors can communicate and compute at the same time [4].

The following notation will be used throughout this paper:

- α_i: The fraction of the entire processing load that is assigned to the i^{th} processor.
- w_i: A constant that is inversely proportional to the computation speed of the i^{th} processor.
- z_i: A constant that is inversely proportional to the channel speed of the i^{th} link
- T_{cp} : The time taken to process a unit load when $w_i = 1$.
- T_{cm} : The time taken to communicate a unit load over link l_i when $z_i = 1$.

In particular, a bus network can be modeled by setting z_i $(i = 1, 2, ..., M)$ in the single level tree network to a common value Z.

III. Expected Time of Searching For A Single record

In this section, the expected time of searching for a record in a dataset distributed over a distributed database is considered. In particular we focus on a linear daisy chain network and a single level tree network. In addition, multi-installment distribution in the single level tree network is considered. All of the processors in the linear daisy chain network and the single level tree network have a front-end processor. That is those processors may compute and communicate simultaneously. The time to send a solution back to processor 0 in the linear daisy chain network or the root processor in the single level tree network is ignored since it is assumed that the solution transmission time is small compared to the load distribution time. It is assumed that all processors will eventually stop at the same time if the desired record is not found. This is the required condition [9] [10] to minimize the time it takes to process the entire dataset in the minimal amount of time (finish time) for the load distribution sequence. Intuitively this must be so as otherwise the finish time could be improved by transferring load from busy to idle processors. Naturally, the record, if present, will definitely be found by the finish time.

A. Finish Time

In this section we seek expressions for finish time to be used in later sections.

1. Linear Daisy Chain Network

In a linear daisy chain network p_0 is connected to p_1, which is connected to p_2 and so on. Let p_0 be the left most processor and p_M be the right most processor. Also number the links $1, 2, \cdots, M$ from left to right. Figure 3 is a timing diagram for a linear daisy chain network. Communication appears above the horizontal axis and communication is below it. It can be seen that the processing time $\alpha_i w_i T_{cp}$ of the i^{th} processor equals the transmission time $(1 - \alpha_0 - \alpha_1 - \cdots - \alpha_i) z_{i+1} T_{cm}$, from the i^{th} processor to the $(i+1)^{th}$ processor plus the processing time $\alpha_{i+1} w_{i+1} T_{cp}$, of the $(i+1)^{th}$ processor where α's are the actual fraction of the load assigned to the i^{th} processor. Thus a set of recursive equations can be written:

$$\alpha_i w_i T_{cp} = (1 - \alpha_0 - \alpha_1 - \cdots - \alpha_i) z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp} \qquad i = 0, 1, \cdots, M-1 \qquad (1)$$

Here $(M+1)$ is the number of processors. These equations, along with the normalization equation, $\sum_{i=0}^{M} \alpha_i = 1$, form a system of $(M+1)$ linear equations in $(M+1)$ unknowns. These can be solved computationally by exploiting the recursive nature of the equations as follows [4]: For $i = 0$,

$$\alpha_0 w_0 T_{cp} = (1 - \alpha_0) z_1 T_{cm} + \alpha_1 w_1 T_{cp} \qquad (2)$$

or

$$\begin{aligned} \alpha_1 &= \left[\frac{w_0 T_{cp} + z_1 T_{cm}}{w_1 T_{cp}} \right] \alpha_0 - \frac{z_1 T_{cm}}{w_1 T_{cp}} \\ &= A_1 \alpha_0 + B_1 \end{aligned} \qquad (3)$$

where:

$$A_1 = \frac{w_0 T_{cp} + z_1 T_{cm}}{w_1 T_{cp}} \qquad (4)$$

$$B_1 = -\frac{z_1 T_{cm}}{w_1 T_{cp}} \qquad (5)$$

For $i = 1$:

$$\alpha_1 w_1 T_{cp} = (1 - \alpha_0 - \alpha_1) z_2 T_{cm} + \alpha_2 w_2 T_{cp} \tag{6}$$

Substituting the expressions for α_1 and solving, α_2 is expressed as follows.

$$\alpha_2 = A_2 \alpha_0 + B_2 \tag{7}$$

where:

$$A_2 = \frac{A_1 (w_1 T_{cp} + z_2 T_{cm}) + z_2 T_{cm}}{w_2 T_{cp}} \tag{8}$$

$$B_2 = \frac{B_1 (w_1 T_{cp} + z_2 T_{cm}) - z_2 T_{cm}}{w_2 T_{cp}} \tag{9}$$

Proceeding in this fashion:

$$\alpha_i = A_i \alpha_0 + B_i \tag{10}$$

Each pair (A_i, B_i) can be computed from (A_{i-1}, B_{i-1}) and the processor and link speed parameters. In general, for $i = 3, \cdots, M$:

$$A_i = \frac{A_{i-1} (w_{i-1} T_{cp} + z_i T_{cm}) + \left(1 + \sum_{j=1}^{i-2} A_j\right) z_i T_{cm}}{w_i T_{cp}} \tag{11}$$

$$B_i = \frac{B_{i-1} (w_{i-1} T_{cp} + z_i T_{cm}) + \left(\sum_{j=1}^{i-2} B_j - 1\right) z_i T_{cm}}{w_i T_{cp}} \tag{12}$$

Substituting these into the normalization equation, α_0 is obtained as follows:

$$\alpha_0 = \frac{1 - \sum_{j=1}^{M} B_j}{1 + \sum_{j=1}^{M} A_j} \tag{13}$$

Thus, the optimal processing time is:

$$T_{finish}(M) = \alpha_0 w_0 T_{cp} \tag{14}$$

$$= w_0 T_{cp} \cdot \frac{1 - \sum_{j=1}^{M} B_j}{1 + \sum_{j=1}^{M} A_j} \tag{15}$$

2. Single Installment Distribution in Single Level Tree Network

The root processor, located at level 0 distributes a fraction of the dataset to each of its children processors, each of which searches for a record in its allocated fraction. Let the links and children processors be numbered $1, 2, \cdots, M$. Each processor at level 1 will search for a record as soon as it receives the entire single installment dataset fraction from the root processor. Figure 4 contains the timing diagram of communication and computation for a single level tree network with single installment load distribution. Let the children processors be numbered $1, 2, \cdots, M$. For Figure 4, the corresponding recursive load distribution equations are

$$\alpha_i w_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp} \qquad i = 0, 1, \cdots, M-1 \tag{16}$$

and the normalizing equation is:

$$\sum_{i=0}^{M} \alpha_i = 1 \tag{17}$$

Equation (16) can be rewritten as

$$\alpha_i = \left(\frac{w_{i+1} + z_{i+1} \cdot \delta}{w_i} \right) \cdot \alpha_{i+1} \tag{18}$$

where:

$$\delta = \frac{T_{cm}}{T_{cp}} \tag{19}$$

These recursive equations can be solved by expressing all the α_i $(i = 0, 1, \cdots, M-1)$ in terms of α_M as

$$\alpha_i = R_i \cdot \alpha_M \tag{20}$$

where:

$$R_i = \prod_{j=i}^{M-1} \left(\frac{w_{j+1} + z_{j+1} \cdot \delta}{w_j} \right) \qquad i = 0, 1, \cdots, M-1 \tag{21}$$

From equation (17), the value of α_M is obtained as:

$$\alpha_M = \frac{1}{1 + \sum_{i=0}^{M-1} R_i} \tag{22}$$

Thus, the fraction of the processing load assigned to the m^{th} processor is:

$$\alpha_m = \frac{\prod_{j=m}^{M-1} \left(\frac{w_{j+1} + z_{j+1} \cdot \delta}{w_j} \right)}{1 + \sum_{i=0}^{M-1} \prod_{j=i}^{M-1} \left(\frac{w_{j+1} + z_{j+1} \cdot \delta}{w_j} \right)} \tag{23}$$

From Figure 4, it can be seen that the finish time $T_{finish}(M)$ is also the processing time of the root processor given by $\alpha_0 w_0 T_{cp}$.

$$T_{finish}(M) = \left(\frac{\prod_{j=0}^{M-1} \left(\frac{w_{j+1} + z_{j+1} \cdot \delta}{w_j} \right)}{1 + \sum_{i=0}^{M-1} \prod_{j=i}^{M-1} \left(\frac{w_{j+1} + z_{j+1} \cdot \delta}{w_j} \right)} \right) w_0 T_{cp} \tag{24}$$

3. Multi-Installment Distribution in Single Level Tree Network

The expected time of searching for a record is expected to decrease as the finish time become smaller. Sending the load fraction in more than one installment so that a processor can begin its computation earlier in time will make the finish time and the time to start searching smaller [5].

Consider a system in which the load is distributed in N installments. The installments are such that all the processors will nominally stop computing at the same time instant. Let $\alpha_{i,j}$ be the fraction of the j^{th} installment assigned to the i^{th} processor and α_i be the fraction assigned to the i^{th} processor.

$$\alpha_i = \sum_{j=1}^{N} \alpha_{i,j} \tag{25}$$

The timing diagram is shown in Figure 5, from which the recursive equations are expressed as follows:

$$\alpha_{i,N} w_i T_{cp} = \alpha_{i+1,N} \left(z_{i+1} T_{cm} + w_{i+1} T_{cp} \right) \qquad i = 1, 2, \cdots, M-1 \tag{26}$$

$$\alpha_{i,j} w_i T_{cp} = \left[\sum_{l=i+1}^{M} \alpha_{l,j} z_l + \sum_{l=1}^{i} \alpha_{l,j+1} z_l \right] T_{cm} \tag{27}$$

$$i = 1, 2, \cdots, M; \qquad j = 1, 2, \cdots, N-1$$

$$\alpha_0 w_0 T_{cp} = \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_{i,j} z_i T_{cm} + \alpha_{M,N} w_M T_{cp} \tag{28}$$

The normalizing equation is given by

$$\alpha_0 + \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_{i,j} = 1 \tag{29}$$

Note that in equation (27) the first summation in the right hand side vanishes when $i = M$.

For convenience, let:

$$\gamma_0 = \alpha_0 \tag{30}$$

$$\gamma_{i,j} = \alpha_{i,j} \cdot z_i \tag{31}$$

Using equation (31), equations (26), (27), (28) and (29) are rewritten as follows:

$$\gamma_{i,N} \frac{w_i}{z_i} T_{cp} = \gamma_{i+1,N} \cdot \frac{1}{z_{i+1}} \left(z_{i+1} T_{cm} + w_{i+1} T_{cp} \right) \qquad i = 1, 2, \cdots, M-1 \tag{32}$$

$$\gamma_{i,j} \frac{w_i}{z_i} T_{cp} = \left[\sum_{l=i+1}^{M} \gamma_{l,j} + \sum_{l=1}^{i} \gamma_{l,j+1} \right] T_{cm} \tag{33}$$

$$i = 1, 2, \cdots, M; \qquad j = 1, 2, \cdots, N-1$$

$$\gamma_0 w_0 T_{cp} = \sum_{i=1}^{M} \sum_{j=1}^{N} \gamma_{i,j} T_{cm} + \gamma_{M,N} \frac{w_M T_{cp}}{z_M} \tag{34}$$

$$\gamma_0 + \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{\gamma_{i,j}}{z_i} = 1 \tag{35}$$

or

$$\gamma_{i,N} = \gamma_{i+1,N} \cdot \left(\sigma_i + \frac{\sigma_i}{\sigma_{i+1}} \right) \qquad i = 1, 2, \cdots, M-1 \tag{36}$$

$$\gamma_{i,j} = \left[\sum_{l=i+1}^{M} \gamma_{l,j} + \sum_{l=1}^{i} \gamma_{l,j+1}\right] \sigma_i \tag{37}$$
$$i = 1, 2, \cdots, M; \qquad j = 1, 2, \cdots, N-1$$

$$\gamma_0 = \sum_{i=1}^{M} \sum_{j=1}^{N} \gamma_{i,j}\sigma_0 + \gamma_{M,N}\frac{\sigma_0}{\sigma_M} \tag{38}$$

$$\gamma_0 + \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{\gamma_{i,j}}{z_i} = 1 \tag{39}$$

Here, the parameter σ_i is the ratio of the communication delay to the computation time of a unit load for i^{th} link and processor.

$$\sigma_i = \frac{z_i T_{cm}}{w_i T_{cp}} \tag{40}$$

For further simplification the following notation is used:

$$\gamma_{i,j} = \gamma_{M,N} R_k \tag{41}$$

Here:

$$k = i + (j-1)M \tag{42}$$

Now, equation (36), (37) and (38) are expressed in terms of $\gamma_{M,N}$ as follows:

$$\gamma_{i,N} = \gamma_{M,N} \cdot \prod_{l=i}^{M-1} \left(\sigma_l + \frac{\sigma_l}{\sigma_{l+1}}\right) \qquad i = 1, 2, \cdots, M-1 \tag{43}$$

$$\gamma_{i,j} = \gamma_{M,N} \left[\sum_{l=i+1}^{M} R_{l+(j-1)M} + \sum_{l=1}^{i} R_{l+jM}\right] \sigma_i \tag{44}$$
$$i = 1, 2, \cdots, M; \qquad j = 1, 2, \cdots, N-1$$

$$\gamma_0 = \gamma_{M,N} \sum_{i=1}^{M} \sum_{j=1}^{N} R_{i+(j-1)M}\sigma_0 + \gamma_{M,N}\frac{\sigma_0}{\sigma_M} \tag{45}$$

In equation (44) and (45), there are two summations. Here these two summations are reduced to one summation. Using (42), equation (44) and (45) are rewritten as follows:

$$\gamma_{i,j} = \gamma_{M,N}\left[\sum_{l=i+(j-1)M+1}^{M+(j-1)M} R_l + \sum_{l=1+jM}^{i+jM} R_l\right]\sigma_i \tag{46}$$

$$= \sigma_i \cdot \sum_{l=k+1}^{k+M} R_l \tag{47}$$

$$i = 1, 2, \cdots, M; \qquad j = 1, 2, \cdots, N-1$$

and

$$\gamma_0 = \gamma_{M,N}\left(\sum_{k=1}^{MN-1} R_k + 1\right)\sigma_0 + \gamma_{M,N}\frac{\sigma_0}{\sigma_M} \tag{48}$$

$$= \gamma_{M,N}\left(\sigma_0 R + \frac{\sigma_0}{\sigma_M}\right) \tag{49}$$

where:

$$R = \sum_{i=1}^{MN-1} R_i + 1 \tag{50}$$

Also $R_{MN} = 1$. Then, from equation (43) and (47), R_k can be expressed as follows:

$$R_k = \prod_{l=k}^{MN-1}\left[\sigma_l + \frac{\sigma_l}{\sigma_{l+1}}\right], \qquad k = (N-1)M+1, \cdots, MN-1 \tag{51}$$

$$R_k = \sigma_{\left[\mathrm{mod}\left(\frac{k-1}{M}\right)+1\right]} \cdot \sum_{l=k+1}^{k+M} R_l, \qquad k = 1, 2, \cdots, (N-1)M \tag{52}$$

Here, $\mathrm{mod}(k/M)$ is the remainder after k is divided by M.

Substituting equation (41) and (49) into equation (39):

$$\gamma_{M,N}\left[\left(\sigma_0 R + \frac{\sigma_0}{\sigma_M}\right) + \sum_{k=1}^{MN-1} \frac{R_k}{z_{\left[\mathrm{mod}\left(\frac{k-1}{M}\right)+1\right]}} + \frac{1}{z_M}\right] = 1 \tag{53}$$

Then $\gamma_{M,N}$ can be obtained as follows:

$$\gamma_{M,N} = \frac{1}{\left(\frac{1}{z_M} + \sum_{k=1}^{MN-1} \frac{R_k}{z_{\left[\mathrm{mod}\left(\frac{k-1}{M}\right)+1\right]}} + \sigma_0 R + \frac{\sigma_0}{\sigma_M}\right)} \tag{54}$$

Thus, from the Figure 5, the finish time $T_{finish}(M,N)$ is given by:

$$T_{finish}(M,N) = \alpha_0 w_0 T_{cp} \tag{55}$$

$$= \gamma_0 w_0 T_{cp} \tag{56}$$

$$= \gamma_{M,N}\left(\sigma_0 R + \frac{\sigma_0}{\sigma_M}\right) w_0 T_{cp} \tag{57}$$

$$= \left[\frac{\sigma_0 R + \frac{\sigma_0}{\sigma_M}}{\frac{1}{z_M} + \sum_{k=1}^{MN-1} \frac{R_k}{z_{\left[\mathrm{mod}\left(\frac{k-1}{M}\right)+1\right]}} + \sigma_0 R + \frac{\sigma_0}{\sigma_M}}\right] w_0 T_{cp} \tag{58}$$

B. Time to Start Searching

Defining the time for each processor to start searching for a record is important since this gives the lower limit of the amount of time till a record is found on a given processor. Let S_m be the time for the m^{th} processor to start searching. Then this time is given in table 1.

Table 1: Time to start searching

Model	S_0	$S_m\ (m = 1, \cdots, M)$
Linear Daisy Chain Network	0	$\sum_{j=0}^{m-1}\left[\left(1 - \sum_{k=0}^{j}\alpha_k\right)\cdot z_{j+1}T_{cm}\right]$
Single Level Tree Network($N = 1$)	0	$\sum_{j=1}^{m}\alpha_j \cdot z_j T_{cm}$
Single Level Tree Network($N \geq 2$)	0	$\sum_{j=1}^{m}\alpha_{j,1} \cdot z_j T_{cm}$

C. Expected Time

In order to completely specify the expected time of searching, one must identify the random variable that describes the record position in the dataset. Here, a record position variable is described in terms of the uniform distribution and is denoted as $\mathbf{X}$, where:

$$\mathbf{X} \sim U(0,1) \tag{59}$$

Since the dataset is normalized, a record is positioned between zero and one. Further, it is assumed that the dataset is very large. Thus, the distribution of a record position is regarded as continuous variable. Now, considering the quantity that must be described as the amount of time till a record is found on a given processor, this random variable denoted by $\mathbf{Y}$ depends on and can be obtained from the random variable of a record position, $\mathbf{X}$.

$$\mathbf{Y} = g(\mathbf{X}) \tag{60}$$

Here, $g(\mathbf{X})$ is the transformation function from $\mathbf{X}$ to $\mathbf{Y}$.

Also $\mathbf{Y}_m$, defined as the distribution of a record in time when the record is found in the m^{th} processor can be obtained from $\mathbf{X}$ using the individual transformation function $g_m(\mathbf{X})$.

$$\mathbf{Y}_m = g_m(\mathbf{X}) \tag{61}$$

where $g_m(\mathbf{X})$ depends only on w_m, the inverse speed of the m^{th} processor. For the originating processor, the transformation function is described as follows:

$$g_0(\mathbf{X}) = \mathbf{X} w_0 T_{cp} + S_0 \qquad 0 \leq \mathbf{X} \leq \alpha_0 \tag{62}$$

For the m^{th} processor, the transformation function, $g_m(\mathbf{X})$ $(1 \leq m \leq M)$ is as follows:

$$g_m(\mathbf{X}) = \left(\mathbf{X} - \sum_{i=0}^{m-1} \alpha_i\right) w_m T_{cp} + S_m \qquad \sum_{i=0}^{m-1} \alpha_i < \mathbf{X} \leq \sum_{i=0}^{m} \alpha_i \tag{63}$$

Here, S_m is the time for the m^{th} processor to start searching as before. Figure 6 and Figure 7 depicts this relationship between $\mathbf{X}$ and $\mathbf{Y}$. For the linear network and single level, single installment tree network the transformation function of Figure 6 maps the record position (a continuous variable from 0 to 1) into being found on a one of the M+1 processors. Similarly the transformation function of Figure 7 maps the record position into a processor for multi-installment distribution or a single level tree network. In other words, the transformation

function, $g(\mathbf{X})$ is the sum of the $g_m(\mathbf{X})$ defined as the distribution of a record in time when the record is found in m^{th} processor.

$$g(\mathbf{X}) = \sum_{m=0}^{M} g_m(\mathbf{X}) \tag{64}$$

From equations (61), (62) and (63), the probability density function of $\mathbf{Y}_m$, denoted as $\mathrm{f}(\mathrm{y}|p_m)$, is obtained as:

$$\mathrm{f}(\mathrm{y}|p_m) = \frac{1}{T_{finish}(M) - S_m} \qquad S_m < \mathrm{y} \leq T_{finish}(M) \tag{65}$$

It is also assumed to be uniformly distributed on $(S_m, T_{finish}(M))$.

Assuming that a single record exists in the dataset, this record exists at one of the $(M+1)$ processors with the probability which is equal to its fraction of the normalized dataset. In other words, if the record is in the fraction assigned to the m^{th} processor, its probability equals to α_m.

$$\Pr\{\text{a record exists at } p_m\} = \alpha_m \tag{66}$$

Now, from the definition of the expected value, the expected time of searching for a record can be written as:

$$\mathrm{E}[\mathbf{Y}] = \int_{0}^{T_{finish}(M)} \mathrm{y} \cdot \mathrm{f}(\mathrm{y}) d\mathrm{y} \tag{67}$$

From Bayes' Theorem, the probability density function of $\mathbf{Y}$ can be expressed as:

$$\mathrm{f}(\mathrm{y}) = \sum_{m=0}^{M} \mathrm{f}(\mathrm{y}|p_m) \cdot \Pr\{\text{a record exists at } p_m\} \tag{68}$$

$$= \sum_{m=0}^{M} \mathrm{f}(\mathrm{y}|p_m) \cdot \alpha_m \tag{69}$$

Here, the probability that a record exists at p_m is already defined and equals to α_m. From

equation (69), equation (67) can be rewritten as:

$$\mathrm{E}[\mathbf{Y}] = \int_0^{T_{finish}(M)} \mathrm{y} \cdot \left[\sum_{m=0}^{M} \mathrm{f}(\mathrm{y}|p_m) \cdot \alpha_m \right] d\mathrm{y} \tag{70}$$

$$= \sum_{m=0}^{M} \alpha_m \left[\int_0^{T_{finish}(M)} \mathrm{y} \cdot \mathrm{f}(\mathrm{y}|p_m) d\mathrm{y} \right] \tag{71}$$

From equation (65), $\mathrm{f}(\mathrm{y}|p_m)$ is only defined on $(S_m, T_{finish}(M))$. Thus the lower bound of the integral on the right side of equation (71) can be substituted as S_m instead of as 0:

$$\int_0^{T_{finish}(M)} \mathrm{y} \cdot \mathrm{f}(\mathrm{y}|p_m) d\mathrm{y} = \int_{S_m}^{T_{finish}(M)} \mathrm{y} \cdot \mathrm{f}(\mathrm{y}|p_m) d\mathrm{y} \tag{72}$$

Solving the integral of the right hand side of the above equation by substituting equation (65) yields:

$$\int_{S_m}^{T_{finish}(M)} \mathrm{y} \cdot \mathrm{f}(\mathrm{y}|p_m) d\mathrm{y} = \frac{[T_{finish}(M)]^2 - [S_m]^2}{2} \cdot \frac{1}{T_{finish}(M) - S_m} \tag{73}$$

$$= \frac{T_{finish}(M) + S_m}{2} \tag{74}$$

Therefore the closed form of the expected time of searching for a record is:

$$\mathrm{E}[\mathbf{Y}] = \sum_{m=0}^{M} \alpha_m \frac{T_{finish}(M) + S_m}{2} \tag{75}$$

Intuitively, this equation holds that the average record search time is the weighted sum of the mid point of each segment weighted by the size of the segment.

IV. Equivalent Expected Time of Searching for Multiple records

The expected time of searching for a single record was considered above. The solution of the expected time of searching for multiple records quickly becomes unmanageable since the last record to appear in the dataset is not always the last record to be detected. This occurs

because the processors shift through the data concurrently changing the order in which detection are made. However, in the case of a single processor, the last record is detected last. Furthermore, since this is a linear model, the expected time of finding *all* of the records can be obtained by applying the mean of the last record position.

$$\mathrm{E}\left[\mathbf{Y}_L\right] = g\left(\mathrm{E}\left[\mathbf{X}_L\right]\right) \tag{76}$$

Here, $\mathrm{E}[\mathrm{X}_L]$ is the mean of the last record position in the normalized dataset.

A multiprocessor will be modeled as a single equivalent processor. For the case of a single equivalent processor denote for convenience the transformation function as $g_{eq}(\cdot)$. Accordingly let:

$$\mathbf{Y} = g_{eq}\left(\mathbf{X}\right) \tag{77}$$

Furthermore, since it is clear that $g_{eq}(\cdot)$ depends on w_{eq}, the inverse speed of the equivalent processor, one can use the equation (62) as follows:

$$g_{eq}\left(\mathbf{X}\right) = \mathbf{X}w_{eq}T_{cp} + S_{eq} \tag{78}$$

Here, S_{eq} is the quasi start time of the equivalent processor. For a single processor, there is no communication delay. However, in the equivalent single processor, the quasi start time should be defined as virtual communication delay which occurs and is preserved in transforming a multi-processor to an equivalent single processor. The general relation between $\mathbf{X}$ and $\mathbf{Y}$ is given below by combining the last two equations is:

$$\mathbf{Y} = \mathbf{X}w_{eq}T_{cp} + S_{eq} \tag{79}$$

Since a multi-processor is considered as a single equivalent processor, the conditional probability density function in equation (65) can be directly applied to probability density function

of $\mathbf{Y}$.

$$\mathrm{f(y)} = \frac{1}{T_{finish}(M) - S_{eq}} \qquad S_{eq} < \mathrm{y} \leq T_{finish}(M) \tag{80}$$

By the definition, the expected time of $\mathbf{Y}$ is derived as follows:

$$\begin{aligned} \mathrm{E}[\mathbf{Y}] &= \int_{S_{eq}}^{T_{finish}(M)} \mathrm{y} \cdot \mathrm{f(y)dy} && (81) \\ &= \int_{S_{eq}}^{T_{finish}(M)} \mathrm{y} \frac{1}{T_{finish}(M) - S_{eq}} \mathrm{dy} && (82) \\ &= \frac{S_{eq} + T_{finish}(M)}{2} && (83) \end{aligned}$$

To obtain S_{eq}, let equation (83) equal to equation (75). Then:

$$S_{eq} = \sum_{m=0}^{M} \alpha_m \left(T_{finish}(M) + S_m\right) - T_{finish}(M) \tag{84}$$

Taking the expectation of both sides of equation (79) yields:

$$\mathrm{E}[\mathbf{Y}] = \mathrm{E}[\mathbf{X}] w_{eq} T_{cp} + S_{eq} \tag{85}$$

The expected value of the distribution $\mathbf{X}$ is $\mu(=0.5)$ when the distribution $\mathbf{X}$ is uniformly distributed on (0, 1). The speed of the equivalent processor w_{eq} can be expressed as follows using equation (85):

$$w_{eq} = \frac{1}{\mu T_{cp}} \left[\sum_{m=0}^{M} \alpha_m \frac{T_{finish}(M) + S_m}{2} - S_{eq} \right] \tag{86}$$

From equation (84), S_{eq} is known. Therefore equation (86) is rewritten as follows:

$$w_{eq} = \frac{1}{\mu T_{cp}} \left[T_{finish}(M) - \sum_{m=0}^{M} \alpha_m \frac{T_{finish}(M) + S_m}{2} \right] \tag{87}$$

However, equation (79) is still not linearly proportional. To create a linearly proportional equation, change the variable $\mathbf{Y}$ as:

$$\mathbf{Y}' = \mathbf{Y} - S_{eq} \tag{88}$$

Then the relation between $\mathbf{X}$ and $\mathbf{Y}'$ is:

$$\mathbf{Y}' = \mathbf{X} w_{eq} T_{cp} \tag{89}$$

It is linear.

Assuming that L records are distributed uniformly in the normalized data, the expected value of the last record location is [11]:

$$\mathrm{E}[\mathbf{X}_L] = 2\mu \frac{L}{L+1} \tag{90}$$

Similarly, $\mathrm{E}[\mathbf{Y}'_L]$ can be expressed like equation (76).

$$\begin{aligned} \mathrm{E}[\mathbf{Y}'_L] &= \mathrm{E}[\mathbf{X}_L] w_{eq} T_{cp} & (91) \\ &= 2\mu \frac{L}{L+1} w_{eq} T_{cp} & (92) \end{aligned}$$

or

$$\mathrm{E}[\mathbf{Y}_L] = 2\mu \frac{L}{L+1} w_{eq} T_{cp} + S_{eq} \tag{93}$$

Substituting equation (84) and (87) into equation (93), the explicit form of above equation is represented as:

$$\begin{aligned} \mathrm{E}[\mathbf{Y}_L] &= 2\frac{L}{L+1} \left[T_{finish}(M) - \sum_{m=0}^{M} \alpha_m \frac{T_{finish}(M) + S_m}{2} \right] \\ &\quad + \left[\sum_{m=0}^{M} \alpha_m \left(T_{finish}(M) + S_m \right) - T_{finish}(M) \right] \end{aligned}$$

The above equation is the expected time of finding all of L records in the dataset.

V. Record Search Evaluation

The finish time, the expected time (section III) and the equivalent expected time (section IV) in a linear daisy chain network and in a single level tree network including multi-installment

load distribution are shown in Figure 8 and Figure 10, respectively. Clearly, as the number of processors increases, all plots in Figure 8 and Figure 10 are decreased. Comparing these two figures, the record searching time in a single level tree network are better than that in a linear daisy chain network with the same parameters. In Figure 10, as the number of installments, N is increased, one can achieve a faster record search time. In Figure 8 and Figure 10 only one record exists in a dataset. In this case, the expected time and the equivalent time are identical.

Figure 9, 11, 12 and 13 shows a comparison of the simulation results and analytic results for the equivalent expected time. Assuming P records in a dataset, the time to search for each record can be calculated using the transformation function in Figure 6 and Figure 7. The largest time to search among P records is the time to search for all records. The simulation result provides the mean time to search for all records. As shown in these figures, the simulation result is very close to the equivalent expected time.

VI. Conclusion

In this paper, the expected time for single record and multiple record search time was evaluated. Here, a linear daisy chain network and a single level tree network were used as target architectures for illustrative purposes. It should be noted that networks as general as a hypercube network can be analyzed using a spanning tree network load distribution model. Furthermore a general tree network which has more than one level can be transformed into a single level tree network by collapsing parent processors with children processors into equivalent processors starting from the terminal node in order to solve for its expected time.

The methodology of section IV is then applicable. The use of divisible load theory to solve for expected record search time is noteworthy as it leads to a tractable analysis and elegant results. The approach taken here can be used for a variety of interconnection topologies and load distribution schedules.

REFERENCES

[1] W. A. Carpenter, and G. S. Heman, "Comparative Analysis of On-line Data Storage Technologies," Technical Letter 11990, MITRE Corporation, McLean, Virginia 1986

[2] S. Bataineh, T. Hsiung, and T. G. Robertazzi, "Closed Form Solutions for Bus and Tree Networks of Processors Load Sharing a Divisible Job." IEEE Transactions on Computers. 43. 10 (1994): 1184-96

[3] S. Bataineh, and T. G. Robertazzi, "Special Section Correspondence: Bus-Oriented Load Sharing for a Network of Sensor Driven Processors." IEEE Transactions on Systems, Man, and Cybernetics. 21. 5 (1991): 1202-5.

[4] 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.

[5] V. Bharadwaj, D. Ghose, and V. Mani, "Multi-installment Load Distribution in Tree Networks With Delays." IEEE Transactions on Aerospace and Electronic Systems. 31. 2 (1995): 555-66.

[6] Y. C. Cheng, and T. G. Robertazzi, "Distributed Computation with Communication Delay." IEEE Transactions on Aerospace and Electronic Systems. 24. 6 (1988): 700-12.

[7] Y. C. Cheng, and T. G. Robertazzi, "Distributed Computation for a Tree Network with Communication Delays." IEEE Transactions on Aerospace and Electronic Systems. 26. 3 (1990): 511-16.

[8] V. Mani, and D. Ghose, "Distributed Computation in Linear Networks: Closed-Form Solutions." IEEE Transactions on Aerospace and Electronic Systems. 30. 2 (1994): 471-83.

[9] T. G. Robertazzi, "Processor Equivalence for Daisy Chain Load Sharing Processors." IEEE Transactions on Aerospace and Electronic Systems. 29. 4 (1993): 1216-21.

[10] J. Sohn and T. G. Robertazzi, "Optimal Divisible Job Load Sharing for Bus Networks." IEEE Transactions on Aerospace and Electronic Systems. 32.1 (1996): 34-39.

[11] H. L. Harter, "Ordering Statistics and Their Use in Estimation and Testing ," Vols. 1 and 2, Superintendent of Documents, U.S. Government Printing Office, 1969.

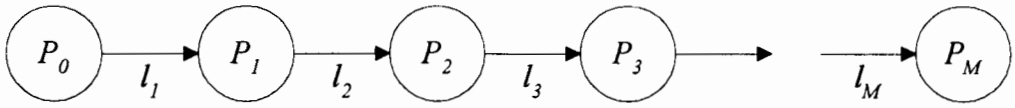

Figure 1: Model of Linear Daisy Chain Network with Communication Links

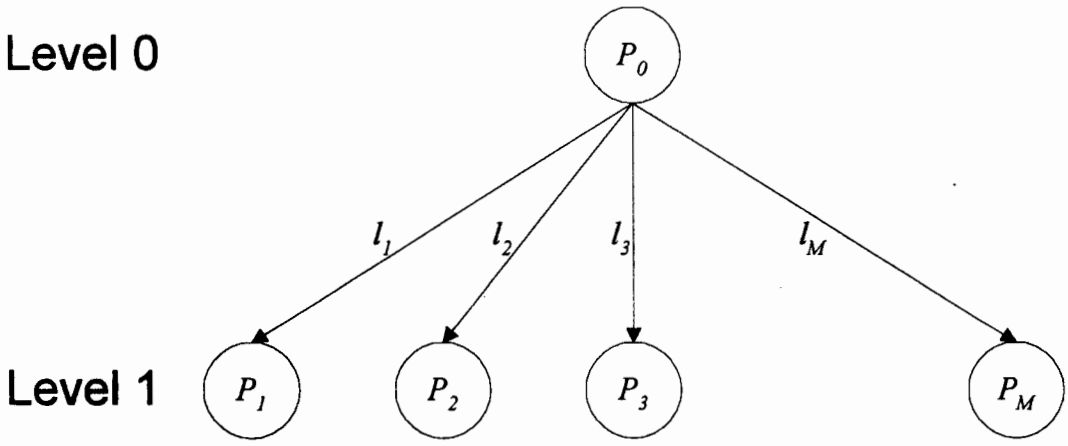

Figure 2: Model of Single Level Tree Network with Communication Links

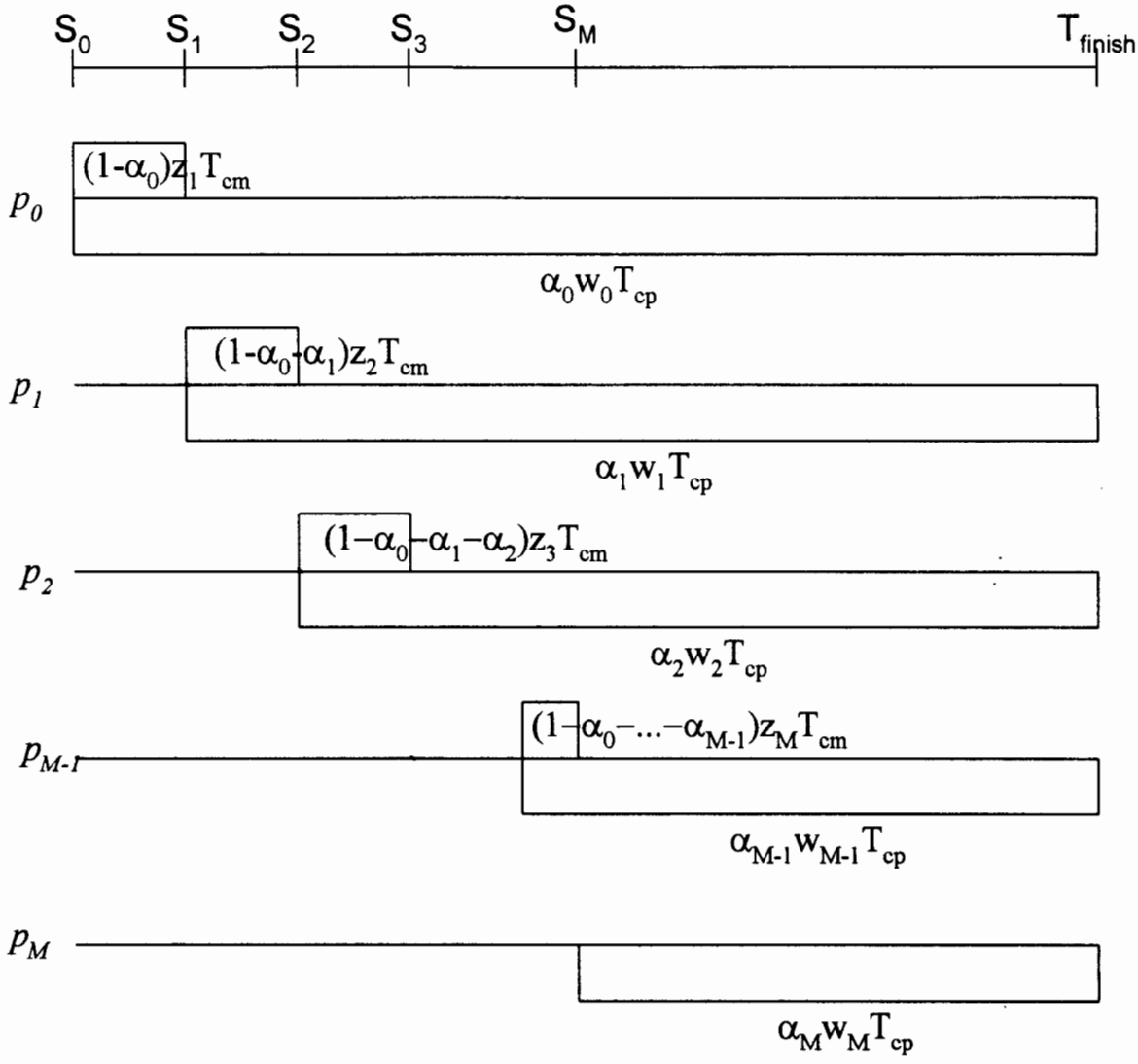

Figure 3: Timing Diagram for Linear Daisy Chain Network

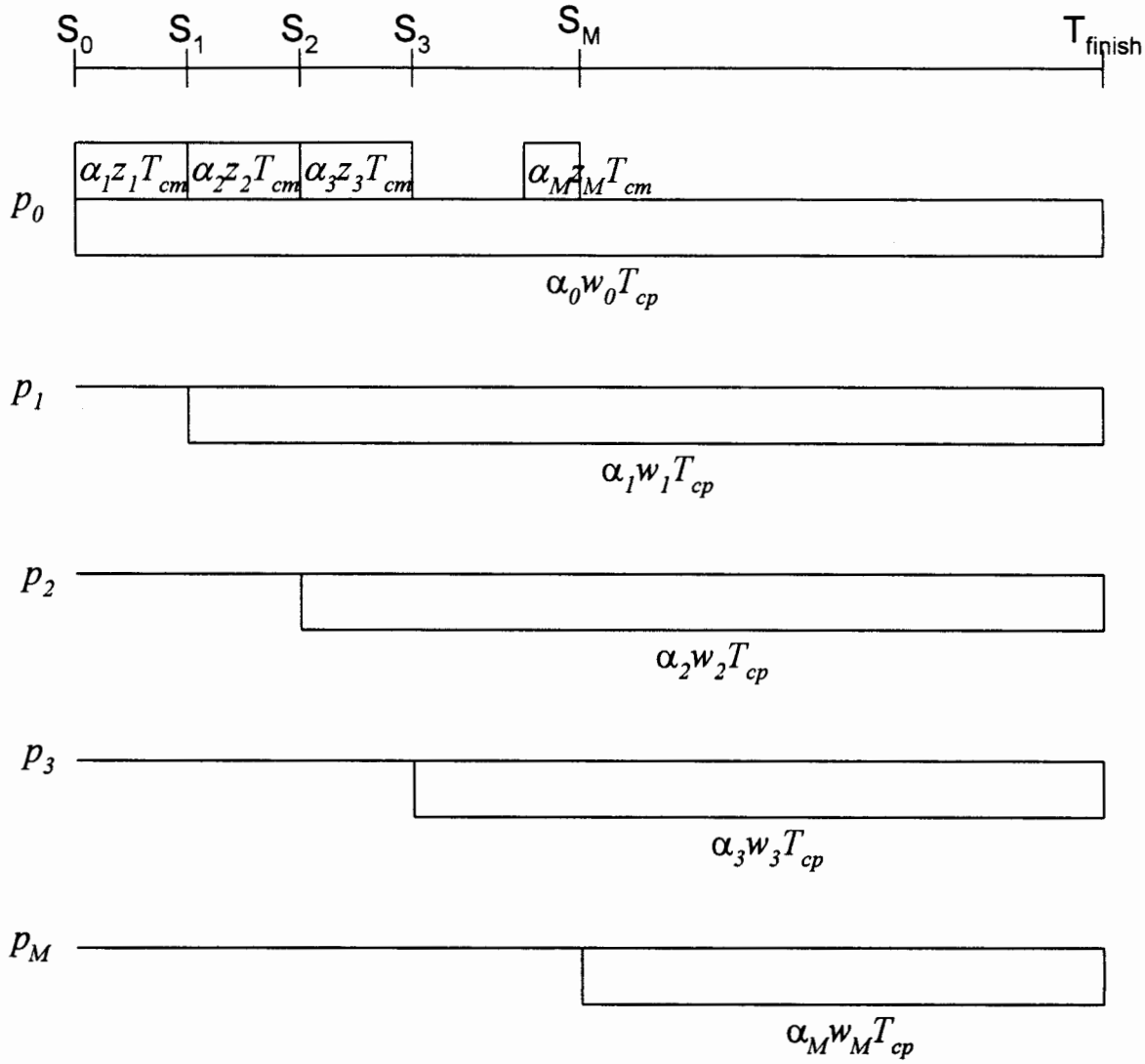

Figure 4: Timing Diagram for Single Installment Distribution in the Single Level Tree Network

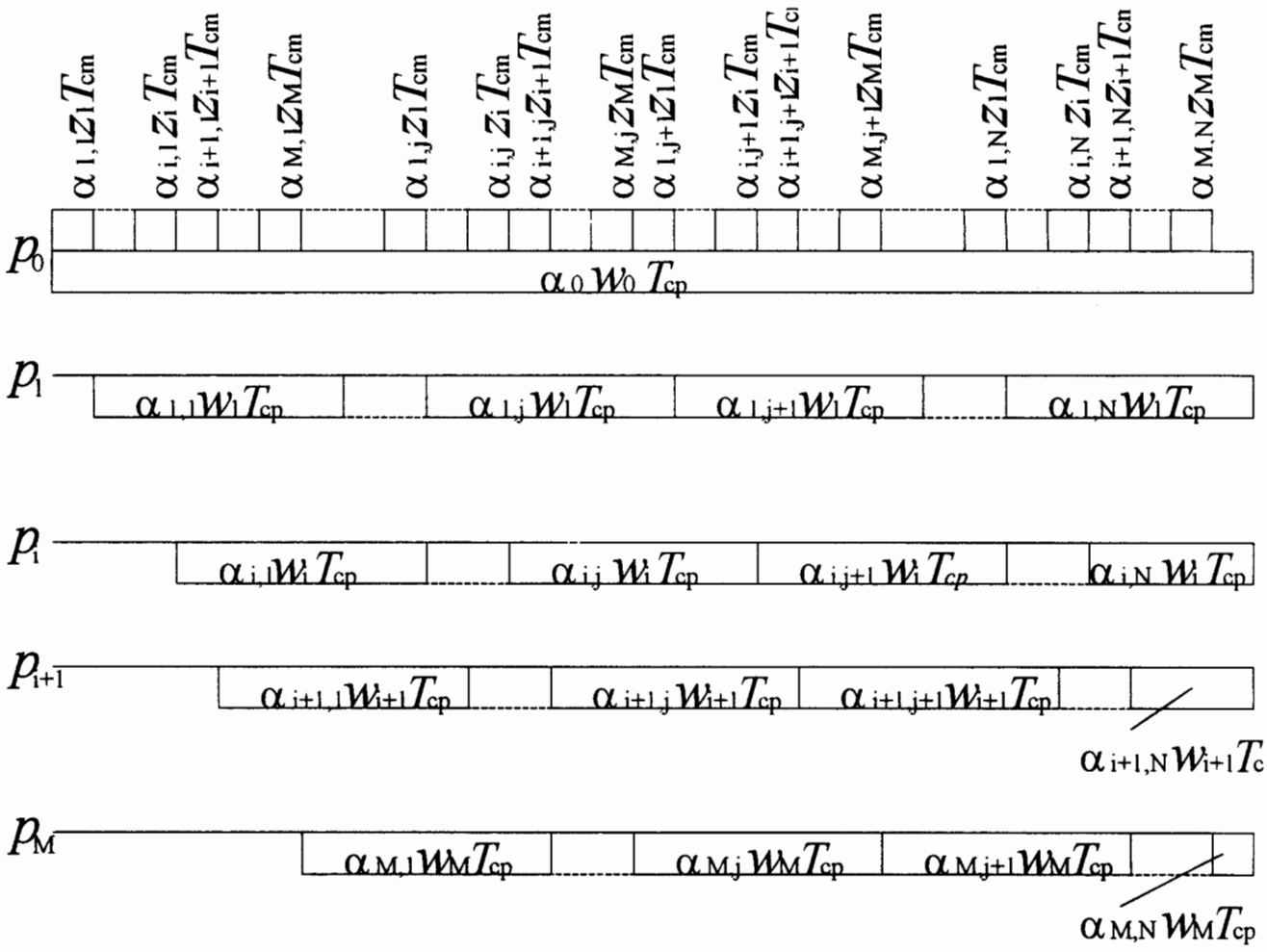

Figure 5: Timing Diagram for Multi-Installment Distribution in the Single Level Network.

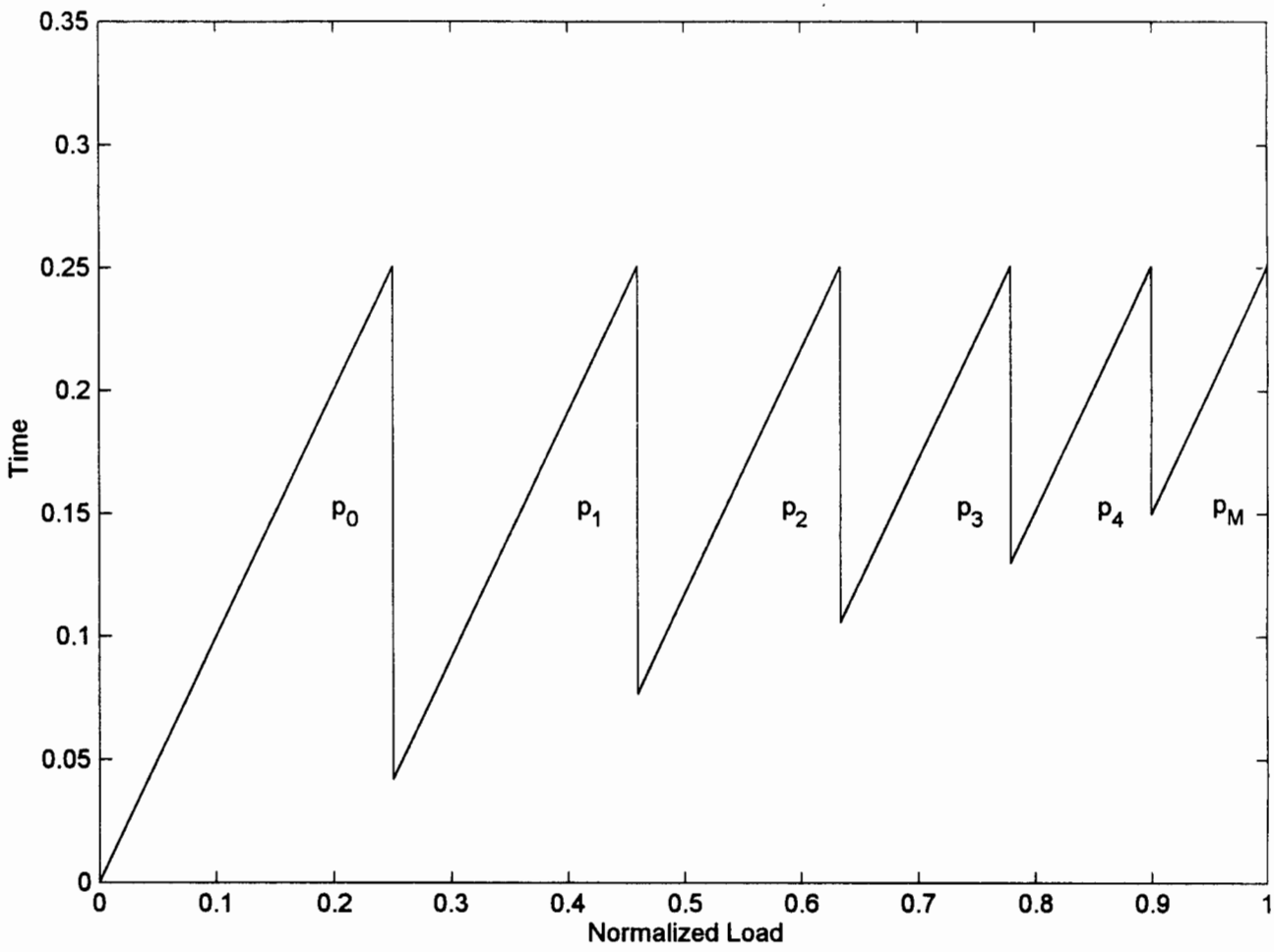

Figure 6: Transformation Function: Linear Daisy Chain Network, Single Level Tree Network; Number of Child Processors, $M = 5$.

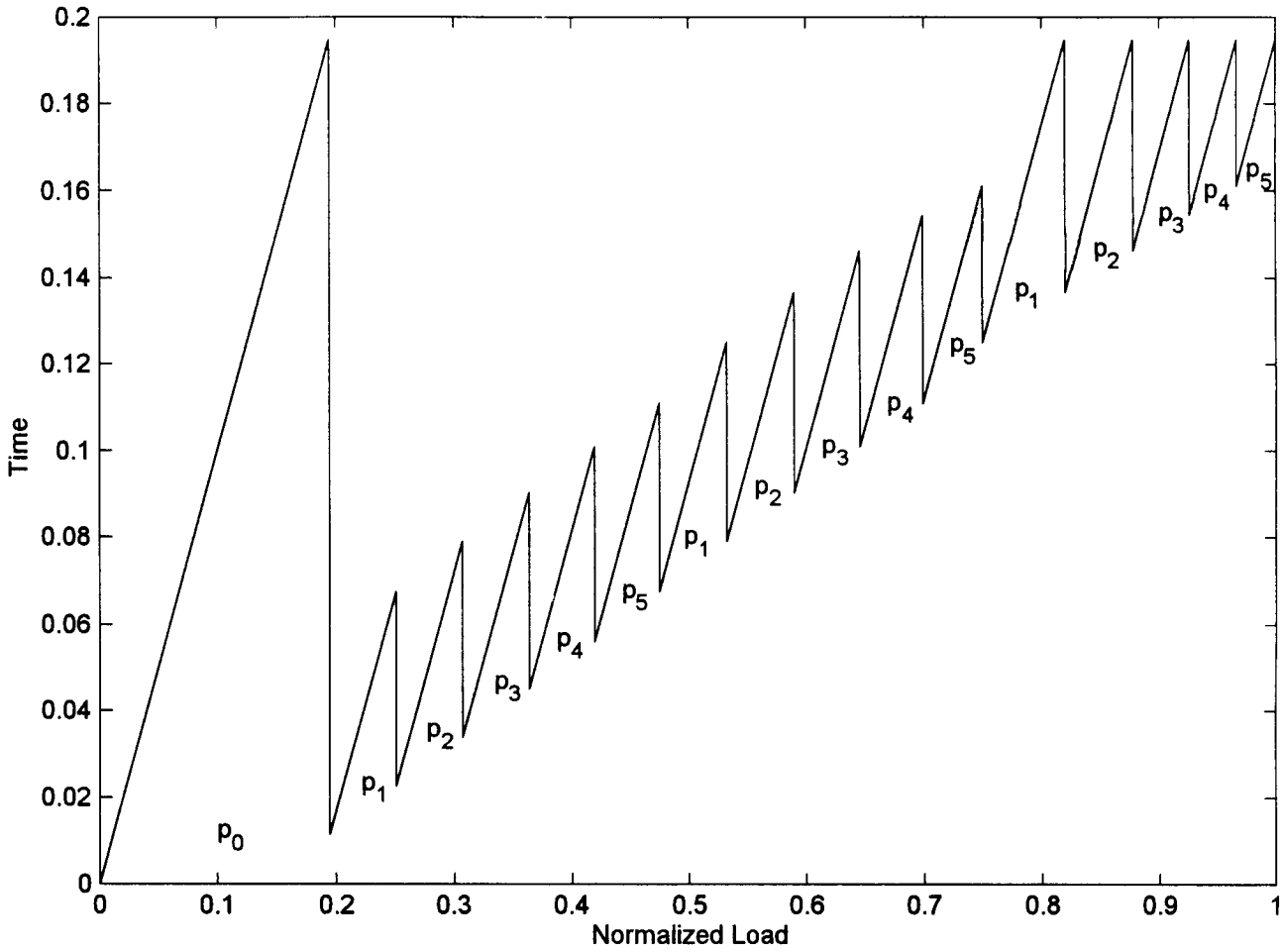

Figure 7: Timing Diagram: Multi-Installment Distribution in Single Level Tree Network; Number of Child Processors, $M = 5$, Number of Installment, $N = 3$.

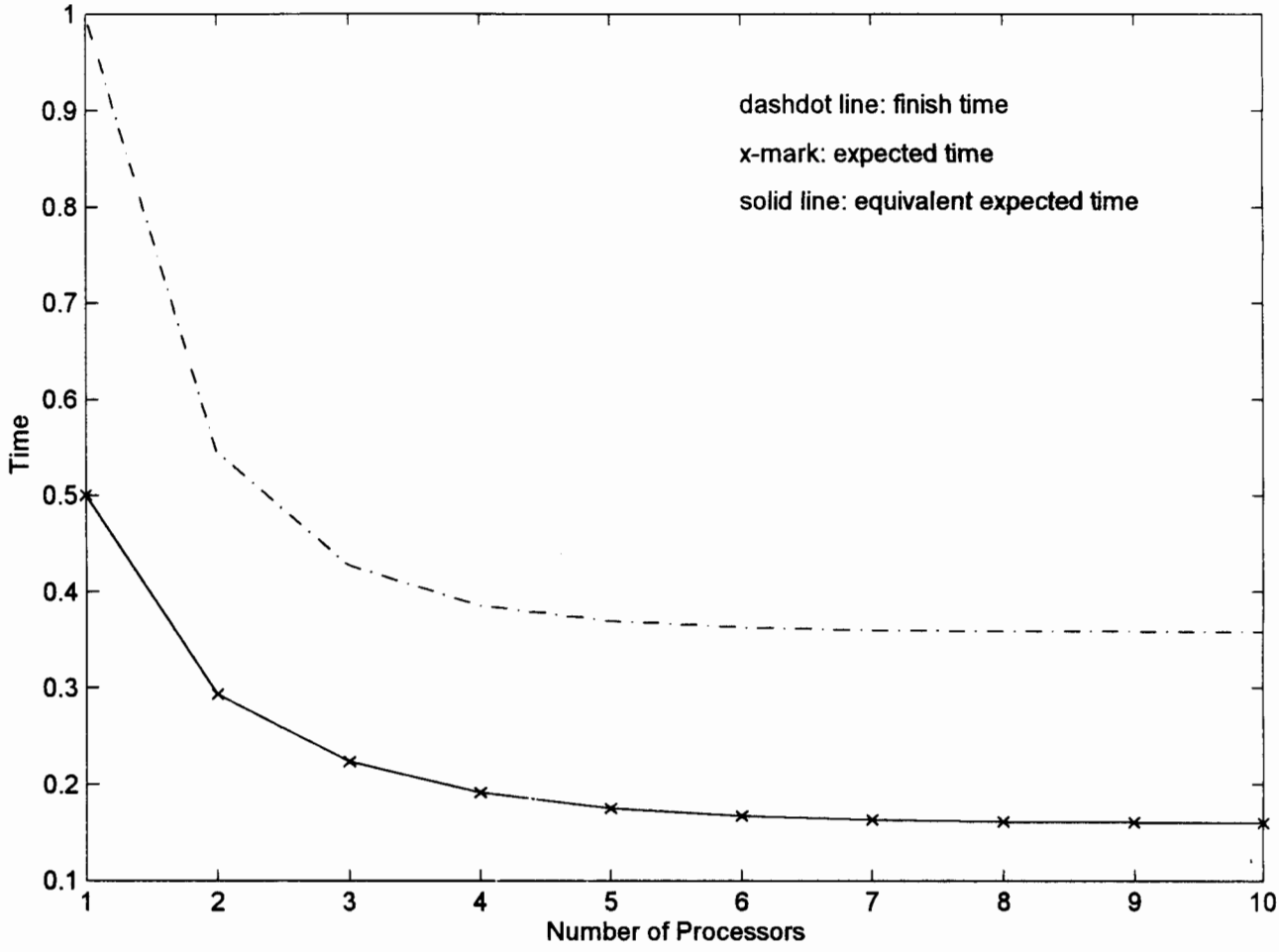

Figure 8: Linear Daisy Chain Network: Time vs. Number of Processors; One Record ; $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$.

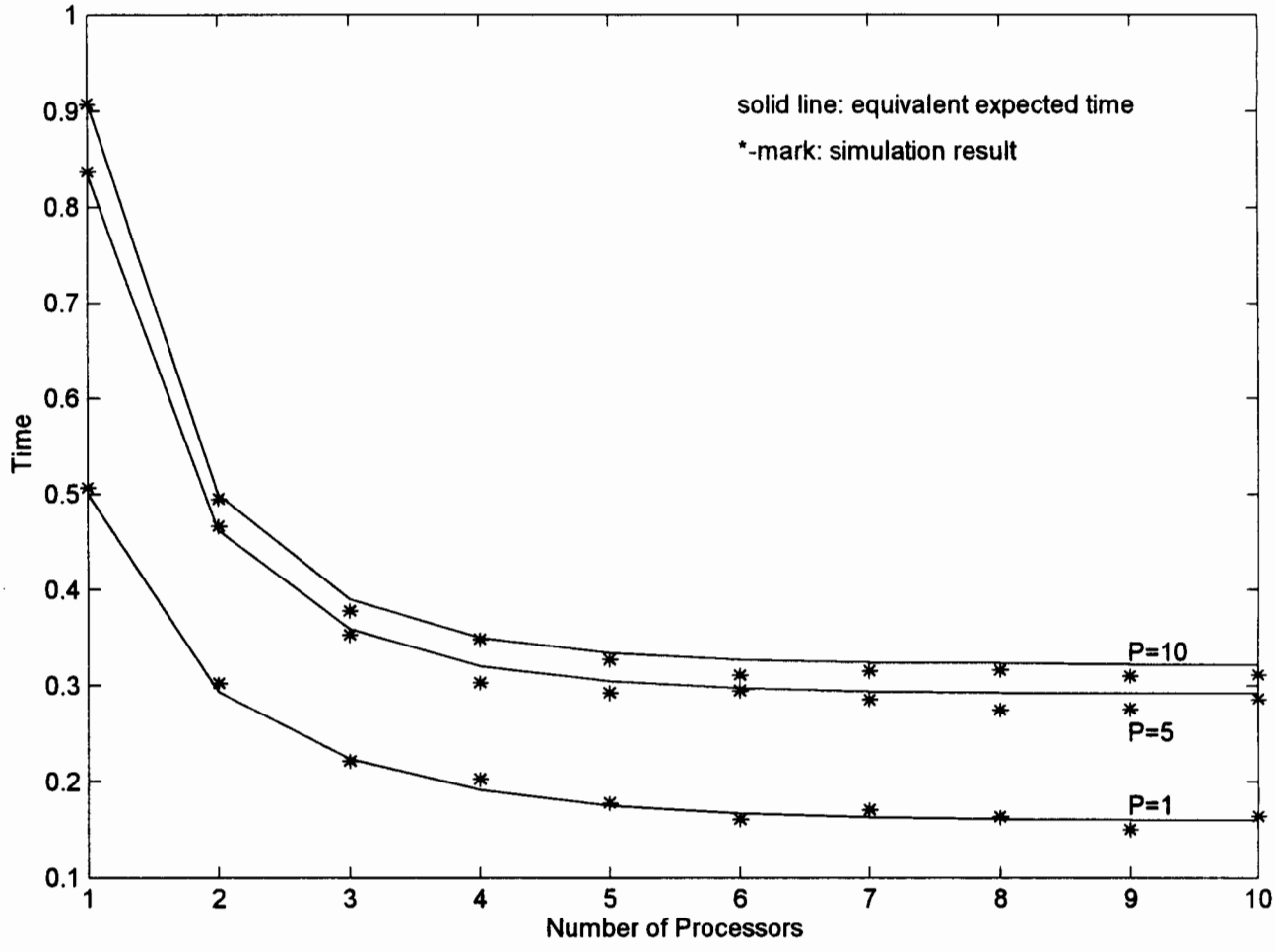

Figure 9: Linear Daisy Chain Network: Time vs. Number of Processors and varying Number of Records, P; $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$.

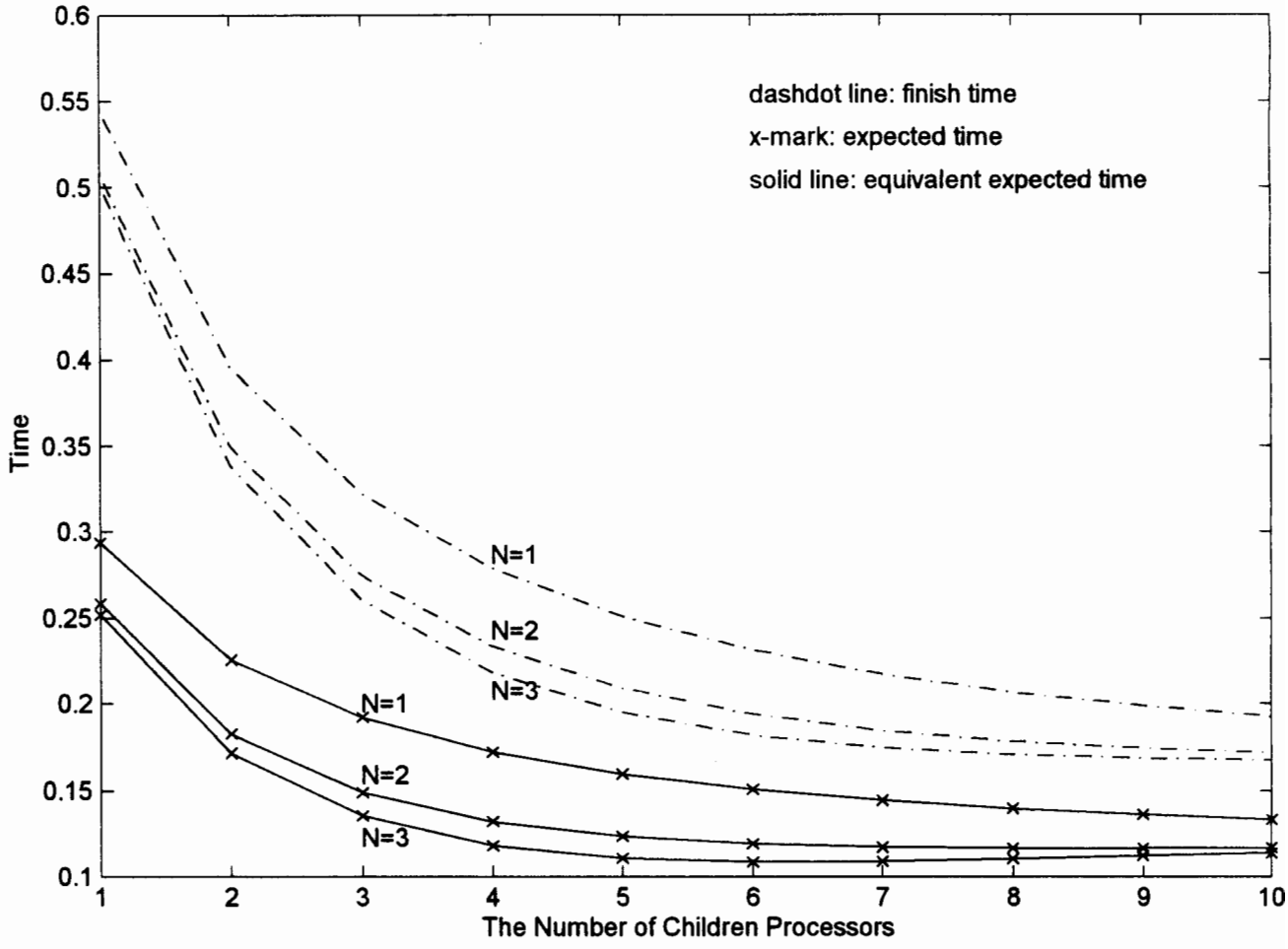

Figure 10: Single Level Tree Network: Time vs. Number of processors and varying Number of Installments, N; One Record; $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$.

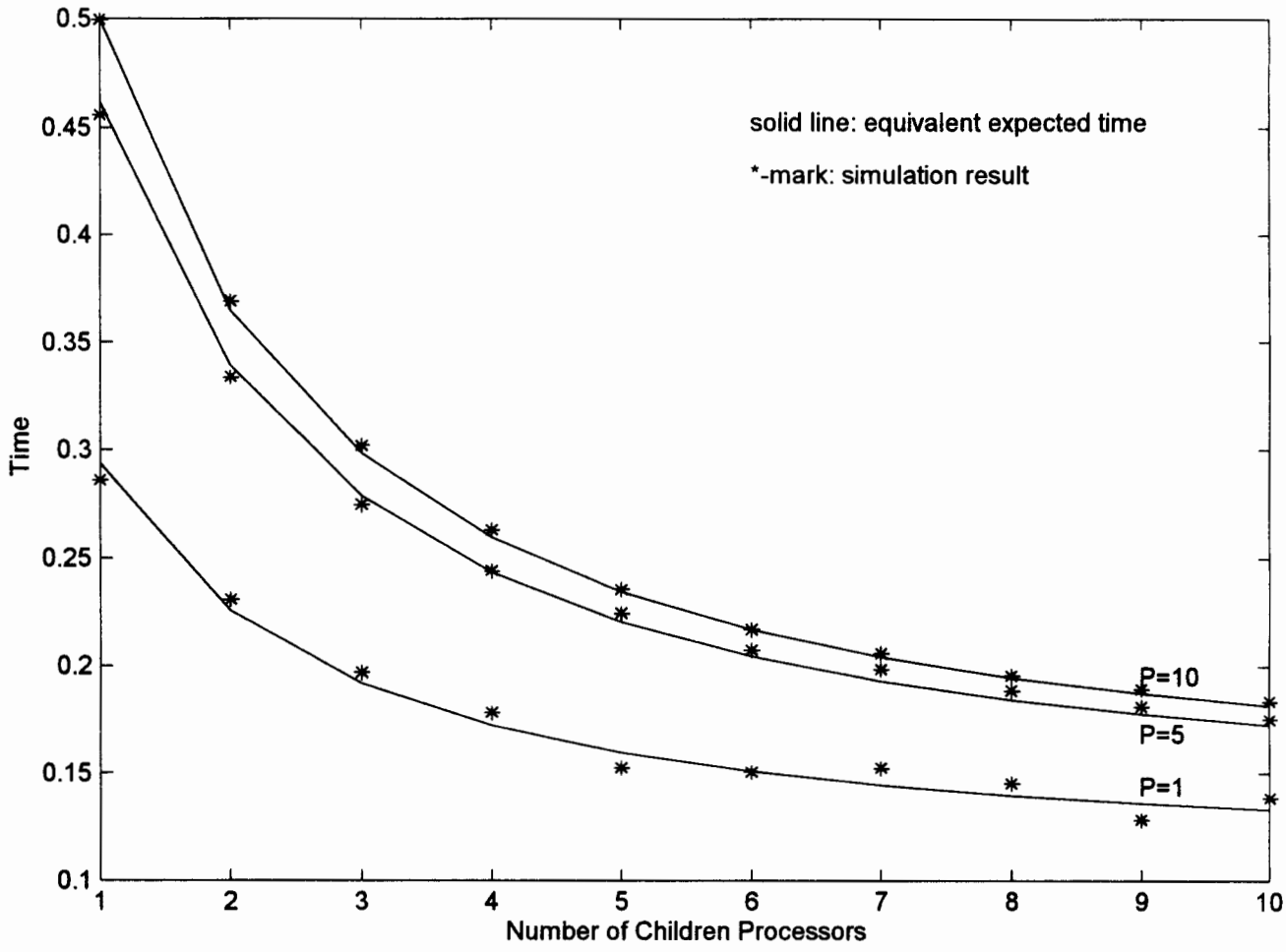

Figure 11: Single Level Tree Network: Time vs. Number of Processors and varying Number of Records, P; One installment ($N = 1$); $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$.

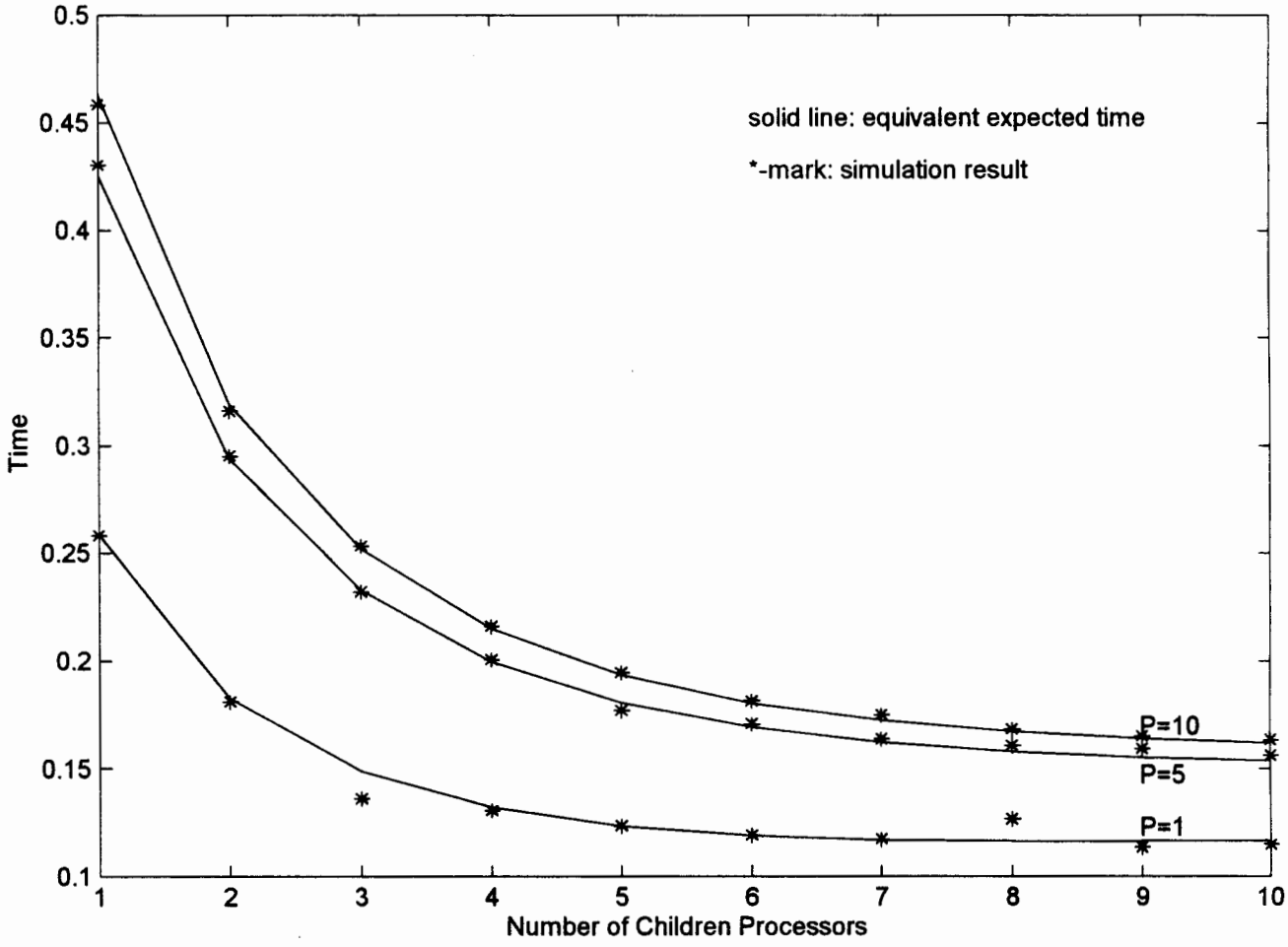

Figure 12: Single Level Tree Network: Time vs. Number of Processors and varying Number of Records, P; Two installments ($N = 2$); $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$.

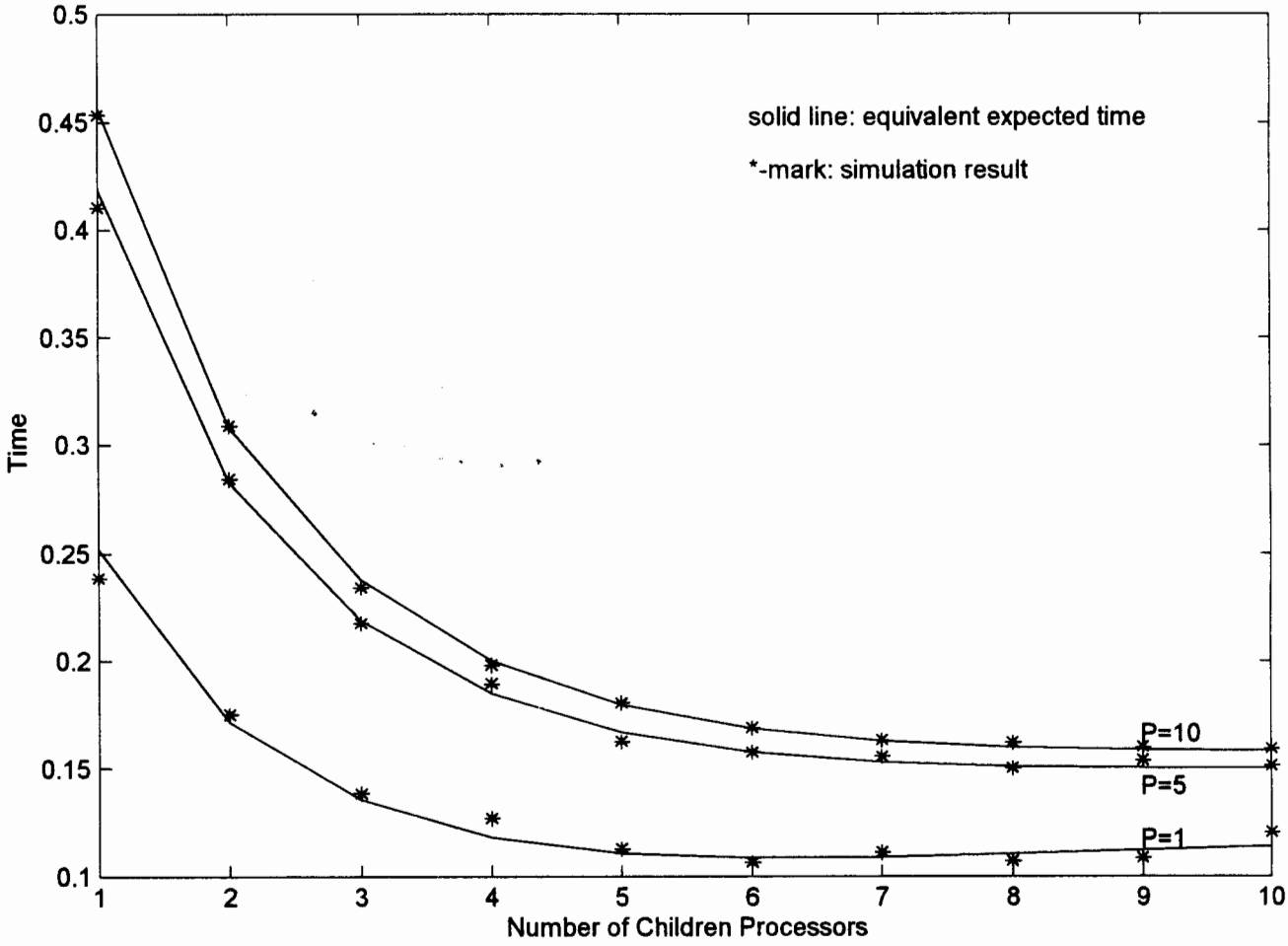

Figure 13: Single Level Tree Network: Time vs. Number of Processors and varying Number of Records, P; Three installments ($N = 3$); $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$.